The Dissertation Committee for Michelle Dawn Swenson
certifies that this is the approved version of the following dissertation:

Phylogenetic Supertree Methods

Committee:

Tandy Warnow, Supervisor

C. Randal Linder, Supervisor

John E. Luecke

Warren A. Hunt Jr.

Lorenzo A. Sadun

Phylogenetic Supertree Methods

by

Michelle Dawn Swenson, B.S. Mathematics

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May 2009

To Lester and Adeline

Acknowledgments

"Me do!" was a phrase my parents often heard from me as a toddler. Though the grammar was wrong, the meaning was clear: don't do it for me, I want to do it myself! To this day my dad still uses this phrase to light-heartedly point out when I am refusing help or advice. He has ample opportunity to mimic this demand from my toddlerdom because my drive to do things for myself and in my own way is still strong. It was, however, my deviations from "me do", allowing others to help and guide me, that got me to where I am today, submitting my dissertation and receiving a Ph.D. in mathematics. I am grateful to the many people throughout my life who have helped me to achieve this goal.

First I thank my parents; by their example, I learned to work hard (but not too hard), to help others, and to pursue the things that bring me joy. My dad is a kind and gentle man with a good-hearted sense of humor and seemingly inexhaustible patience whose presence has always been a comfort. These traits combine to provide an excellent shoulder to cry on, which I've done throughout this challenging time. My mom has had a powerful influence on my educational experience, as an example, when she returned to college and completed her degree during my grade school and junior high years, and as my advocate, arguing on my behalf for my school to allow me to take more challenging math classes. It was her evident pride in (or rather, her bragging about) my math accomplishments that led the undergraduate advisor at UNL to encourage me to apply for the math fellowship

which solidified math as my major area of study. My brother Timm is a reminder that many paths lead to successful and satisfying careers. His, and my parents', example have helped to keep me grounded while completing my degree.

Many friends have paved the long path to this dissertation: Brenna convinced me that I was actually quite intelligent, and planted the seed that I should consider college. Dane let me ride his coattails through a process that landed us in math and physics courses at the University of Nebraska at Lincoln during our senior year of high school. The Math Grrrls, especially Taryne, were with me over marathon weekends of math, and contributed to my belief that girls do math the "right" way, collaboratively! Dorea got me to test the waters of math research and eventually apply to graduate school. The enthusiastically supportive faculty members at UNL, in particular Wendy Hines, Jamie Radcliffe, Judy Walker, Gordon Woodward, and Tom Marley, along with the Math Grrrls, made my undergraduate career an absolute joy. Amanda inspired me to seek out research allowing collaboration with biologists.

My advisors, Tandy Warnow and Randy Linder, made the most tangible contributions to this dissertation, and to my abilities as a researcher. From the beginning, Tandy encouraged me to find and follow my passions. She was patient in the development of our advisor-advisee relationship. Tandy knew herself and what she needed from me as an advisee, and continually made an effort to know me and what I needed in an advisor. As icing on the cake, she provided me amazing turn-around time and feedback on this dissertation, giving most generously of her time and attention. Randy readily answered questions about biology (some many times over, each as patiently as the last). He kept my eye on making tools that are useful, and answering questions of interest, to the biological community. I value his advice on so much of my work and academic life, from biological considerations and programming techniques to teaching and career.

I thank my committee members, John Luecke, Warren Hunt, and Lorenzo

Sadun for their ideas on how to expand the scope of my work and their sound career advice. John Luecke offered a wealth of advice over the years, more than I can possibly include here. Though not on my committee, David Hillis was always available to answer questions, and provide feedback on talks and manuscripts. He lead many of the IGERT program activities I found invaluable in my work.

I've had the privilege of working with many wonderful folks on an array of research endeavors. My collaboration with Mike Steel and Stefan Gruenwald during my semester at Canterbury University in New Zealand was an amazing experience. Without Francois, who wrote much of the code for SMIDGen, and Rahul, who implemented SuperFine, the results I present here would not have been achieved. I thoroughly enjoyed both advising and collaborating with undergraduates Colin, Daniel, and Young. My past and present lab mates, Serita, Kevin, Sindhu, Shruthi, Ganesh, Luay, and Cara, have been excellent sounding boards, and a major reason why my talks have gone so well. Again, Serita, thank you, thank you, thank you! Many outside Tandy's lab have also given me valuable feedback and camaraderie, including Ruth, Tracy, Shannon, Derrick, Jeremy, Kris, Mary, Alan, and Becky.

Many folks have made life here in Austin, Texas beautiful. My best friend, Betseygail, has been so supportive in so many areas of my life. Among the many, many wonderful times we've shared, countless greasy lunches consumed at the Posse East, enjoying the smoldering heat and country music, somehow seem most relevant here. Both Rusty and Deena never fail to brighten my day. I feel a bootcamp-survival-type-bond with many of the math graduate students who went through prelims with me, particularly Jennifer, Brian, and Benni. I have thoroughly enjoyed my involvement with the Graduate Women in Computing, initially as member, ultimately a board member. I have connected with several women in the group, particularly Alison, Maria, and Jenn. I am ever so grateful to Walter for the never-ending supply of chocolate, making many a rough day a little more bearable.

I would not have made it through graduate school without dance classes, but the opportunity to perform as well has been a blessing. My heart swells to think of the wonderful people I have performed with in Austin; in particular my friend Kim, also a graduate student dancing professionally. I am grateful to Arletta and Jennifer, the directors of Austin Dance Ensemble and Austin Classical Ballet, respectively, for bringing me into their groups and providing me this wonderful outlet.

Finally, I thank Chris Larson. I have always heard people use the phrase "words cannot describe" when trying to thank some one. Until now, I could not completely relate to that sentiment. Words really cannot describe the gratitude I have for Chris and the multitude of ways he has helped me through this process. He helped me develop and effectively use a database for the overwhelming volume of data I needed to process for this dissertation. He put his years as an english major to use editing my early papers, and nearly every draft of this dissertation. Besides the countless hours spent coding and editing, Chris gave me continual emotional support, encouraged me, pointed out my progress as a writer (no matter how many times I asked him not to), ran countless errands, made me many dinners, reminded me to take a break, and was wonderful company in those few moments when I was not writing, editing, generating data, or dancing. Most importantly, even though I wasn't always smart enough to take it, he was always there with a loving hug whenever I needed it.

Michelle Dawn Swenson

The University of Texas at Austin
May 2009

Phylogenetic Supertree Methods

Publication No. _____

Michelle Dawn Swenson, Ph.D.
The University of Texas at Austin, 2009

Supervisors: Tandy Warnow, C. Randal Linder

The central task in phylogenetics is to infer the evolutionary relationships among a given set of species. These relationships are usually represented by a phylogenetic tree with the species of interest at the leaves and where the internal vertices of the tree represent ancestral species. The amount of available molecular data is increasing exponentially and, given the continual advances in sequencing techniques and throughput, this explosive growth will likely continue. These vast amounts of available data mean that biologists are able to assemble large multi-gene datasets for use in phylogenetic analyses, which presents distinct computational challenges.

Supertree methods comprise one approach to reconstructing large phyloge-

nies, given estimated trees for overlapping subsets of the entire set of taxa. These source trees are combined into a single supertree on the full set of taxa using various algorithmic techniques. When the data allow, the competing approach is a combined analysis (also known as a "super-matrix" or "total evidence" approach), whereby the different sequence data matrices for each of the different subsets of taxa are put into a single super-matrix, and a tree is estimated on that super-matrix.

In this dissertation, I present simulation software I designed to allow users to compare the relative performance of different supertree methods, as well as that of combined analysis, on more realistic data and on a larger scale than has been used up to this point. I present an extensive simulation study that uses this software to compare the performance of supertree methods and combined analysis, and that demonstrates a need for more topologically accurate supertree methods. I also introduce a new supertree method that I have developed that outperforms the most commonly used, and what until now has arguably been the most accurate, supertree method.

Contents

Chapter 1

Introduction

The study of evolution is fundamental to the investigation of a wide array of biological questions. For example, estimates of the evolutionary history of sets of molecular sequences are used in biomedical research, including drug and vaccine development [e.g., Bush et al., 1999], in tracking the origins and development of humans over time [e.g., Templeton, 1992], and even as forensic evidence in the investigation of criminal acts [e.g., Metzker et al., 2002]. One of the most ambitious goals in evolution is to discover the relationships among all the species on Earth, the Tree of Life. The field responsible for this undertaking is phylogenetics.

How one might go about inferring this tree is a central difficulty when considering the Tree of Life. In a typical molecular phylogenetic analysis, a tree on a particular set of species is constructed by first collecting the DNA or protein sequences for a homologous gene in each species, and then using those sequences to construct a tree on that set of species. (Homology is shared evolutionary history, and genes are called homologous if they descended from a common ancestor gene.) Using this sort of process to construct the Tree of Life is not feasible, since no single gene would suffice. A gene or region that was appropriate for constructing the very deep relationships in the tree would evolve too slowly to resolve relationships closer

to the leaves, and conversely a gene or region that was appropriate for resolving the relationships closer to the leaves would evolve too quickly to recover the deeper relationships. Furthermore, it is unlikely that a single gene would contain enough information to reconstruct the such a large tree.

Since a single gene won't do the trick, what about collecting the sequences from multiple genes? There are two main approaches for analyzing multi-gene datasets: constructing a tree by analyzing the entire dataset as a whole, or constructing a tree on each gene dataset separately and then combining those trees into a single tree on the entire set of species. The former is called the combined analysis, supermatrix, or "total evidence" approach; the latter is called the supertree approach, and a method that combines trees with overlapping leaf sets into a single tree is called a supertree method.

In addition to analyzing multi-gene datasets, supertree methods have several other uses, including 1) summarizing the results of available studies on (subsets of) a particular group of interest when access to the sequences is not possible, 2) producing a tree from disparate data-types, such as molecular, morphological, and gene-order data, that require independent types of analysis (and therefore prohibit a combined analysis) and 3) analyzing large datasets that would take too long to analyze using other phylogenetic reconstruction methods. The use of supertree methods to solve such problems has received much criticism [e.g., Rodrigo, 1993, Slowinski, 1999, Gatesy et al., 2002, Gatesy and Springer, 2004]. For example, many supertree methods have been shown to have a size bias, favoring the relationships supported by larger input trees over smaller ones, [e.g., Purvis, 1995b] some have also been shown to have a shape bias, producing either balanced or unbalanced trees more often [e.g., Wilkinson et al., 2005], and the input trees on which supertrees are constructed may not be completely independent (some primary data may have been used in the inference of more than one input tree)[e.g. Gatesy et al., 2002]. How-

ever, given the limitations of current sequence-based phylogenetic reconstruction methods, supertrees are presently a necessary tool for many phylogenetic problems [Bininda-Emonds et al., 2003]. In fact, most, if not all, detailed estimates of the Tree of Life (e.g. the Tree of Life Web Project; http://tolweb.org/tree/phylogeny.html) to date have been constructed using a supertree approach: using informal supertree methods, splicing together various phylogenetic trees and tree representations of taxonomies, to achieve a single tree on all currently known species.

In Chapter 2, I introduce the basic definitions and concepts needed to discuss the work in the following chapters, and discuss the existing literature that provides a context for this dissertation. In Chapter 3, I present a software platform designed to test the performance of supertree and combined analysis methods. This platform produces more realistic datasets on which to test supertree methods than others currently used. In Chapter 4, I compare the performance of supertree methods against that of combined analyses. I show that, in general, the combined analysis approach is more accurate than, and thus preferable to, the most widely used supertree approach, matrix representation with parsimony (MRP) [Baum, 1992, Ragan, 1992, Baum and Ragan, 2004]. Thus, under circumstances where combine analyses are infeasible or impossible, the results in Chapter 4 highlight the need for more accurate supertree methods.

In Chapter 5, I describe a new supertree method, SuperFine, and show that it outperforms MRP, runs in significantly less time than MRP, and approaches the accuracy of a combined analysis. Prior to the development of SuperFine, no existing supertree method was shown to consistently outperform MRP.

In Chapter 6, I focus on collections of rooted triples and the larger relationships that sets of rooted triples may imply. This chapter contains several theoretical results, including one that has a practical application to a previously proposed supertree method.

Finally, in Chapter 7, I summarize the contributions of this dissertation and discuss their implications. In this chapter I also discuss future directions for the work in this dissertation.

Chapter 2

Background

In Section 2.1, I introduce the basic definitions I will use to discuss phylogenetic trees and supertree methods. In Section 2.2, I present the models of sequence evolution that are integral to many phylogenetic estimation techniques. Section 2.3 describes the steps in a typical phylogenetic analysis. In Section 2.4, I discuss the necessity of simulation studies in assessing the performance of phylogenetic methods, present the steps in a typical simulation study, and describe how simulation studies are used to analyze the performance of supertree methods. Section 2.5 gives a detailed description of MRP, the most commonly used supertree method, and of a variant of that method. In Section 2.6, I discuss works related to the simulation platform presented in Chapter 3.

2.1 Basic Definitions

In studying evolutionary histories, I will refer to particular entities of interest, be they species, genes, molecular sequences, or languages, as *taxa* (singular *taxon*). This is a generalization of the use of the term taxa in biology where a taxon is any of the taxonomic categories such as phylum, order, or species in the Linnean

system. I assume, in general, that the taxa have evolved via a process that can be represented by a rooted tree – a directed acyclic graph, in the graph theoretic sense – whose root represents the most recent common ancestor of all the taxa represented in the tree. At the leaves of the tree are extant taxa (or possibly extinct taxa for which we have fossil records), the internal vertices are ancestral taxa, and each edge represents an ancestor-descendant relationship with the ancestor being the taxon represented by the vertex closer to the root of the tree.

A **phylogenetic tree** is a rooted or unrooted tree with leaves labeled by extant taxa. Internal vertices are usually unlabeled and have degree at least three. A **rooted phylogenetic tree** contains a significant vertex called the "root" having degree two or more. An unrooted phylogenetic tree T is **binary** if all internal vertices have degree three; a rooted phylogenetic tree is binary if all internal vertices have degree three, except the root, which has degree two. In Figure 2.1, T_1 and T_2 are unrooted, T_3, T_4, and T_5 are rooted, T_1, T_3, and T_4 are binary, and T_2 and T_5 are non-binary. Notice that T_1 can be obtained from T_3 or T_4 by suppressing the root vertex of either of these rooted trees. Equivalently, T_3 and T_4 can be obtained from T_1 by subdividing an edge in T_1 with a root vertex. We use $L(T)$, $V(T)$, and $E(T)$ to denote the leaf set, vertex set, and edge set of a phylogenetic tree T, respectively. Discussions of supertrees often concern collections of trees, which we will denote $\mathcal{T}$, and the union of the leaf sets of the members of the collection, which we will denote $L(\mathcal{T})$.

Accurately rooting a phylogenetic tree is a complex problem requiring specific knowledge of the set of taxa being studied or the assumption of a molecular clock. If the molecular data used to reconstruct a phylogeny are assumed to have evolved at a constant rate over time, then one can root the tree based on estimated leaf-to-leaf distances. This assumption is often violated in real datasets. In mathematical (and computational) phylogenetics, therefore, our goal is to reconstruct only the *unrooted*

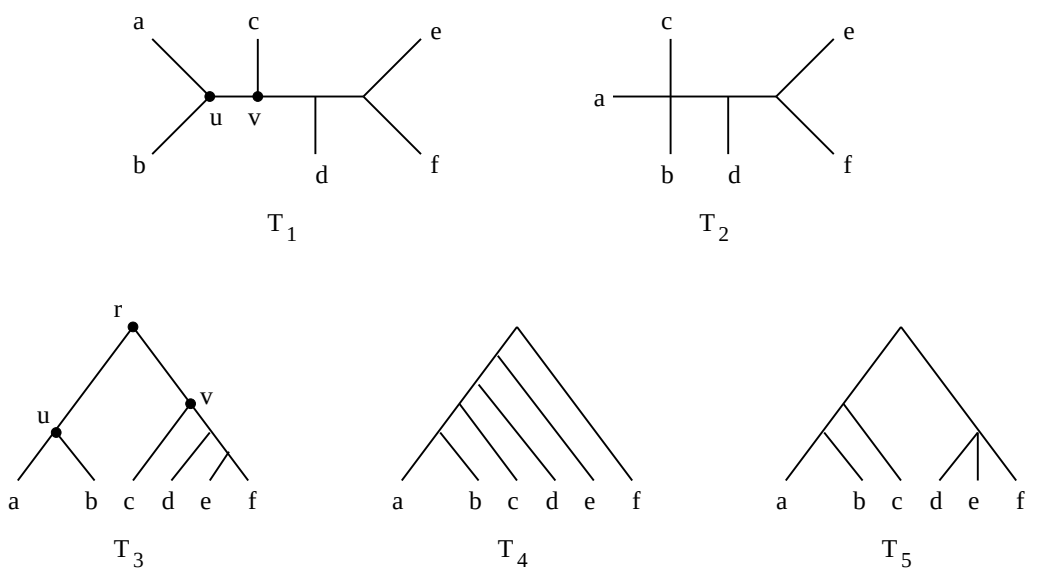

Figure 2.1: T_1 and T_2 are unrooted trees, T_3, T_4, and T_5 are rooted trees, T_1, T_3, and T_4 are binary trees, and T_2 and T_5 are non-binary trees.

version of the rooted tree that represents the evolutionary relationships among the taxa.

2.1.1 Clusters and Bipartitions

Each vertex in a rooted tree defines a group of taxa that are more closely related to each other than they are to any other taxon in the tree; such a group is called a *clade*. The clade defined by a vertex v is the set $C(v)$ containing all the taxa that can be reached from the root via a path containing the vertex v. In mathematical phylogeny, such a group is often called a cluster. The set of all clusters defined by some vertex in a tree T is denoted $\mathcal{H}(T)$; i.e, $\mathcal{H}(T) = \{C(v) : v \in V(T)\}$. Notice that $\mathcal{H}(T)$ is a hierarchy, which means that for every two members C_1, C_2 of $\mathcal{H}(T)$, either $C_1 \subseteq C_2$, $C_2 \subseteq C_1$, or $C_1 \cap C_2 = \emptyset$. The trivial clusters, those contained in every tree with leaf set X, are X (the leaf set itself) and the singleton sets $\{x\}$ where $x \in X$. The vertex v in T_3 of Figure 2.1, for example, defines the

cluster $\{c, d, e, f\}$, and the set of non-trivial clusters defined by the vertices of T_3 is $\{\{a, b\}, \{e, f\}, \{d, e, f\}, \{c, d, e, f\}\}$. We say that the rooted tree T' **refines** T if $\mathcal{H}(T) \subset \mathcal{H}(T')$; this is denoted $T \leq T'$.

The analogous relationships in unrooted trees are bipartitions of the taxon set, and are defined by edges rather than vertices. Each edge e in a phylogenetic tree T, when deleted from T, creates two new subtrees T_1 and T_2. We say that the edge e induces the bipartition $L(T_1)|L(T_2)$, equivalently $L(T_2)|L(T_1)$, of $L(T)$; this bipartition is denoted σ_e. The full set of induced bipartitions of a tree T is denoted $\Sigma(T)$, i.e. $\Sigma(T) = \{\sigma_e : e \in E(T)\}$. The trivial bipartitions, those contained in every unrooted tree with leaf set X, are those involving a single taxon on one side of the bipartition $\{x\}|X - \{x\}$ where $x \in X$. We often abuse notation, dropping the brackets and commas when writing the sets on either side of the split. Thus, we say that in Figure 2.1, the edge (u, v) in T_1 induces the bipartition $ab|cdef$, and the set of non-trivial bipartitions of T_1 is $\{ab|cdef, abc|def, abcd|ef\}$. Naturally, we say that an unrooted tree T' **refines** T if $\Sigma(T) \subseteq \Sigma(T')$; again, this is denoted $T \leq T'$. Note that the definition of $\Sigma(T)$ does not require that T is unrooted, thus $\Sigma(T)$ is still defined for rooted trees. $\mathcal{H}(T)$, on the other hand, does not automatically extend to unrooted trees.

We often compute the distance between two trees based on the difference in their non-trivial bipartitions. For the following discussion, let T be the tree that represents the true phylogenetic relationships for the leafset S, such that $|S| = n$, and let $\hat{T}$ be an estimated tree on the same set of leaves. An edge $e \in E(\hat{T})$ is a **false positive** if $\sigma_e \notin \Sigma(T)$; an edge $e \in E(T)$ is a **false negative** if $\sigma_e \notin \Sigma(\hat{T})$. The **false positive distance** and **false negative distance** of $\hat{T}$ to T are the number of false positives in $\hat{T}$ and false negatives in T, respectively. We denote these distances by $FP(\hat{T}, T)$ and $FN(\hat{T}, T)$. Thus $FP(\hat{T}, T) = |\Sigma(\hat{T}) - \Sigma(T)|$ and $FN(\hat{T}, T) = |\Sigma(T) - \Sigma(\hat{T})|$. The **false positive rate** is simply the false positive

distance normalized by the number of internal branches of $\hat{T}$ and the **false negative rate** is the false negative distance normalized by the number of internal branches of T. If both the true tree and estimated tree are binary then $FP(\hat{T}, T) = FN(\hat{T}, T)$, and the false positive rate equals the false negative rate.

In addition to their use in measuring the distance between two trees, bipartitions can be used to construct a single tree that summarizes the relationships in a collection of trees with identical leafsets. The methods used to construct such summary trees are called *consensus methods*, and the trees produced *consensus trees*; the most commonly used consensus methods are the strict consensus and majority consensus. The strict consensus tree of a set of trees is the tree whose set of bipartitions are those that are present in all the trees in the given set. For a given set of trees $\{T_1, T_2, ..., T_k\}$, the **strict consensus** tree T is such that $\Sigma(T) = \{\sigma : \sigma \in \Sigma(T_i) \text{ for all } 1 \leq i \leq k\} = \bigcap_{i=1}^{k} \Sigma(T_i)$. The majority consensus tree of a set of trees is the tree whose bipartition set contains those bipartitions present in more than half of the trees in the given set. For a given set of trees $\{T_1, T_2, ..., T_k\}$, the **majority consensus** tree T is such that $\Sigma(T) = \{\sigma : \text{the number of trees } T_i \text{ such that } \sigma \in \Sigma(T_i) \text{ is greater than } \frac{k}{2}\}$.

2.1.2 Subtrees and Supertrees

Let T be a phylogenetic tree and let A be a non-empty subset of $L(T)$. The **induced subtree** of T on the leaf set A, denoted $T|_A$, is the tree obtained by first taking a minimal subtree of T induced by the elements of A and then suppressing all vertices of degree two (except the root if T is a rooted tree). The tree T **displays** T' if $L(T') \subseteq L(T)$ and $T' \leq T|_{L(T')}$. Let $\mathcal{T}$ be a collection of phylogenetic trees that are either all rooted or all unrooted; then, T is said to display $\mathcal{T}$ if for all $T' \in \mathcal{T}$, T displays T'. The *compatible span* of $\mathcal{T}$, denoted $co(\mathcal{T})$, is the set of all trees with leaf set $L(\mathcal{T})$ that display $\mathcal{T}$. We say that a collection of trees $\mathcal{T}$ is **compatible** if

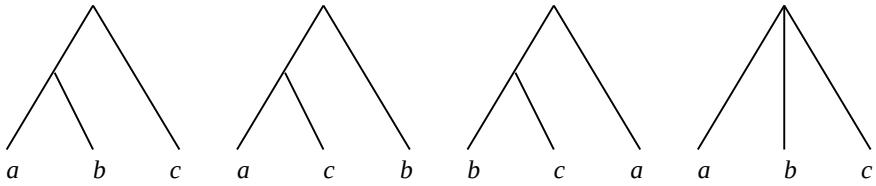

Figure 2.2: The four rooted trees with three leaves.

$co(\mathcal{T})$ is non-empty, and that it is **incompatible** otherwise.

A **supertree method** will be defined as any function that takes as input a collection of phylogenetic trees $\mathcal{T}$, and returns a tree, or a collection of trees, with leaf set $L(\mathcal{T})$. The input trees to supertree methods are called **source trees**, and a tree in the output is often referred to as a **supertree**. Notice that there is no requirement that a supertree method must produce supertrees that contain any information from the source trees other than that of their leaf sets. For example, even if a collection of source trees $\mathcal{T}$ is compatible, a supertree method need not return an element of $co(\mathcal{T})$; such a supertree method, however, is certainly not a desirable one.

2.1.3 Rooted Triples and Quartet Trees

For every three leaves, a, b, c, there are four possible rooted phylogenetic trees with leaf set $\{a, b, c\}$, see Figure 2.2. A rooted binary tree on three leaves is called a **rooted triple**. A rooted triple with leaf set $\{a, b, c\}$ that contains the cluster $\{a, b\}$ is denoted $ab|c$ (equivalently, $ba|c$). The set of rooted triples displayed by a rooted tree T is denoted $r(T)$; formally, $r(T) = \{ab|c : a, b, c \in L(T) \text{ and } \{a, b\} \in \mathcal{H}(T|_{\{a,b,c\}})\}$. In Chapter 6, we will discuss properties of subsets of $r(T)$, as well as functions on such subsets.

For every four leaves, a, b, c, d, there are four possible unrooted phylogenetic trees with leaf set $\{a, b, c, d\}$, Figure 2.3. An unrooted binary tree with four leaves

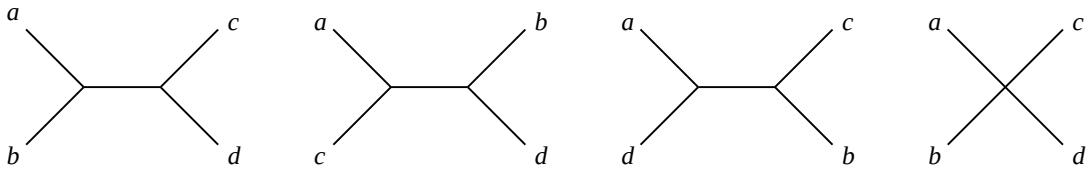

Figure 2.3: The four unrooted trees with four leaves.

is called a **quartet tree**. A quartet tree with leaf set $\{a, b, c, d\}$ and non-trivial bipartition $ab|cd$ is denoted by that bipartition (equivalently, $ba|cd$, $ab|dc$, $ba|dc$, or $cd|ab$, etc.). The set of quartet trees displayed by an unrooted tree T is denoted $q(T)$; formally, $q(T) = \{ab|cd : a, b, c, d \in L(T) \text{ and } ab|cd = T|_{\{a,b,c,d\}}\}$. In Chapter 5, we will present a supertree method that is partially based on such sets of quartet trees.

2.2 Models of Sequence Evolution

Stochastic models of site evolution in a nucleic or amino acid sequence assume sequences evolve down a tree starting with a sequence at the root. As this sequence evolves down the tree it undergoes a series of point mutations, or substitutions. While such models can be described for many types of character data, we will restrict our discussion in this section to DNA sequence evolution. In this case, a site substitution is a single nucleotide mutating to another nucleotide. Under pure substitution models of sequence evolution, more complex mutations such as insertions, deletions, duplications, and inversions are not allowed. Under these models, if the sequence at the root had length l, then the each sequence that results at a leaf of the tree also has length l. Most models of sequence evolution assume, further, that each site evolves independently of, and identically to, all other sites; this is called the *i.i.d.* (identically and independently distributed) assumption. Under such a model, we only need to describe the model of evolution for a single site.

An edge weighted tree (T, w) is a tree topology T together with an edge-weighting function $w : E(T) \rightarrow \mathbb{R}^+$, where $w(e)$ typically represents the number of expected changes per site on a the edge e, Besides requiring a model tree (T, w) the model also requires defining an instantaneous substitution rate matrix.

The description of models of sequence evolution that follows is adapted from Swofford et al. [1996, pages 432-444]. An instantaneous rate matrix M is defined to be a matrix in which each element M_{ij} represents the rate of change from base i to base j during some infinitesimal time period dt. For DNA sequences, this is a 4×4 matrix

$$M = \begin{pmatrix} -(a\pi_C + b\pi_G + c\pi_T) & a\pi_C & b\pi_G & c\pi_T \\ g\pi_A & -(g\pi_A + d\pi_G + e\pi_T) & d\pi_G & e\pi_T \\ h\pi_A & j\pi_C & -(h\pi_A + j\pi_C + f\pi_T) & f\pi_T \\ i\pi_A & k\pi_C & l\pi_G & -(i\pi_A + k\pi_C + l\pi_G) \end{pmatrix},$$

where $a, b, c, ...,$ and l are the relative rates at which one base transforms into another, and $\pi_A, \pi_C, \pi_G,$ and π_T are frequency parameters that correspond to the equilibrium frequencies of the bases A,C, G, and T, respectively. The diagonal elements of M are always chosen so the elements in the corresponding row sum to zero. This is called the General Markov (GM) model.

A model is said to be time-reversible if for all pairs of bases i, j, $M_{ij} = M_{ji}$. Most models of sequence evolution used in phylogenetic analyses are time-reversible. The most general time-reversible model (GTR) has instantaneous rate matrix

$$M = \begin{pmatrix} -(a\pi_C + b\pi_G + c\pi_T) & a\pi_C & b\pi_G & c\pi_T \\ a\pi_C & -(a\pi_C + d\pi_G + e\pi_T) & d\pi_G & e\pi_T \\ b\pi_G & d\pi_G & -(\pi_G(b+d) + f\pi_T) & f\pi_T \\ c\pi_T & e\pi_T & f\pi_T & -\pi_T(c+e+f) \end{pmatrix}$$

In this dissertation I refer to two common sub-models of GTR, the Jukes-Cantor model (JC) [Jukes and Cantor, 1969] and Kimura's 2-parameter model (K2P) [Kimura, 1980]. The Jukes-Cantor model assumes equal base frequencies ($\pi_A = \pi_C = \pi_G = \pi_T = \frac{1}{4}$), and that all substitutions from one base to another occur at

the same rate (a=b=c=d=e=f=1). Thus, the rate matrix for JC is

$$M = \begin{pmatrix} -\frac{3}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & -\frac{3}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & -\frac{3}{4} & \frac{1}{4} \\ -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{3}{4} \end{pmatrix}$$

Kimura's two-parameter model allows transitions (A↔G or C↔T) to occur at a rate different than transversions (A↔C, A↔T, C↔G, or G↔T). Thus, for K2P, $a = c = d = f = 1$, and $b = e = \kappa$, resulting in the rate matrix

$$M = \begin{pmatrix} -\frac{1}{4}(\kappa+2) & \frac{1}{4} & \frac{1}{4}\kappa & \frac{1}{4} \\ \frac{1}{4} & -\frac{1}{4}(\kappa+2) & \frac{1}{4} & \frac{1}{4}\kappa \\ \frac{1}{4}\kappa & \frac{1}{4} & -\frac{1}{4}(\kappa+2) & \frac{1}{4} \\ -\frac{1}{4} & \frac{1}{4}\kappa & \frac{1}{4} & -\frac{1}{4}(\kappa+2) \end{pmatrix}$$

Rate heterogeneity across sites can be incorporated into these models of sequence evolution without violating the assumption that the sites are *i.i.d.* by multiplying the instantaneous rate matrix by a rate drawn from a distribution. There are two commonly used distributions. The simpler of the two is a discrete distribution, called an invariable-sites model, that assumes some fraction p of the sites is incapable of undergoing substitutions, and that the remaining sites draw their rates from some common distribution. The addition of such an assumption to a given model is indicated by appending the model name with "+I", GTR+I, for example, and is specified by selecting a value for the parameter p, in addition to M and (T, w). The second, more complex distribution used to incorporate rate heterogeneity is the Γ distribution with both the shape parameter and the scale parameter set to a single value α. The addition of the assumption of Γ distributed rates across sites is indicated with "+Γ", GTR+Γ, for example, and requires specifying a value

for the parameter α. In the simulation study presented in Chapter 4, we evolve sequences under a GTR+Γ+I model. When both Γ and I are modeled, each site has a probability p of being invariable. When a site is allowed to vary, it then selects its rate from the Γ distribution with shape parameter α.

2.2.1 Identifiability

This discussion of the identifiability of models of evolution is adapted from Linder and Warnow [2006, page 14]. Suppose that a character (or site in a set of sequences) has r states and there are n taxa. (For DNA sequences, $r = 4$.) Then there are r^n possible ways the character can assign states to the n leaves, and these are the "patterns" for the character.

A model of evolution is defined by the tree topology T over which the sequences evolve, and set of parameters Θ that are required to describe how those sequences evolve over T. For the JC model, Θ consists of only of a edge-weighting function w; for the GTR model, Θ consists of the edge weights and an instantaneous rate matrix M. For a given model of evolution $\mathcal{M}$, T and Θ, determine the probabilities of each pattern occurring at the leaves of T. Thus, for a given model, T and Θ define a probability distribution on the r^n possible patterns at the leaves.

Definition 2.2.1. *Let $\mathcal{M}$ be a model of sequence evolution over an r-state alphabet. Then $\mathcal{M}$ is **identifiable** if for any $(T, \Theta) \in \mathcal{M}$, T and Θ can be recovered based solely on the probability distribution on the r^n patterns, where $n = L(T)$.*

All of the models described so far in this section are identifiable. However, this is not true for all stochastic models. An example of a model that is not identifiable is the "no-common mechanism" model (NCM) [Steel and Penny, 2000], which assumes a common instantaneous rate matrix for all sites but allows the rate for each individual site and each edge to be unique.

2.2.2 Statistical Consistency

The following discussion of statistical consistency is adapted from Kim and Warnow [1999]. One measure of the quality of a phylogenetic reconstruction method is the topological accuracy of the unrooted leaf-labeled tree obtained by the method.

Definition 2.2.2. *Let $\mathcal{R}$ be a phylogenetic reconstruction method and $\mathcal{M}$ be a model of sequence evolution. Then is $\mathcal{R}$ is* ***statistically consistent*** *under the model $\mathcal{M}$, if for any $(T, \Theta) \in \mathcal{M}$ and any $\epsilon > 0$, there exists a sequence length l, for which given sequences of length at least l, generated under the model (T, Θ), $\mathcal{R}$ will return the tree T with probability at least $(1 - \epsilon)$.*

2.3 The Process of a Phylogenetic Study

When conducting a molecular phylogenetic analysis, biologists first determine the scope of their study, choosing which phylogenetic group they will study, which particular taxa they will include in the study, and what genetic marker or markers they will sequence to conduct their study. (Some studies may, instead, use morphological characters of the taxa, such as physical characteristics or biological functions, but I will focus on studies that use molecular data.) The limitations that govern which taxa are included in the study include availability of tissue, location of specimens, time, funding, etc., and the ability to amplify and sequence.

Phase 1: Data collection

Biologists go out into the field and collect specimens (organisms or tissues) of the taxa they have decided to study and make use of museum and herbarium specimens. The goal of the data collection phase is to assemble, for each taxon, a homologous sequence or sequences of characters that will be used to construct a phylogenetic tree on the set of taxa they have chosen to study. For studies using molecular characters,

these data are molecular sequences for the particular marker they have decided to use in the study. To obtain these sequences, they extract DNA (or RNA) from their specimens and sequence the desired genetic marker. Following sequence assembly, the remainder of the phylogenetic analysis is primarily computational.

Phase 2: Sequence Alignment

An alignment is constructed taking the putatively homologous sequences and putting them in a matrix such that the sequence from each individual taxon occupies its own row, and each column of nucleotides have ideally evolved from a common ancestral nucleotide. This object property of the columns is called positional homology. Positional homology is estimated by inserting gaps into each sequence in the appropriate places, and padding the beginning and end of each sequence with place holders that represent unknown nucleotides, so as to obtain sequences having the same length. For most phylogenetic reconstruction methods, positional homology is of critical importance because each column of the alignment is assumed to represent the independent evolutionary history of the site. An incorrect alignment will have two or more sites that do not reflect positional homology. Therefore, at least some of the data used to reconstruct the phylogeny will suggest a false evolutionary history of the sequences. To obtain an alignment, most researchers use some software, such as ClustalW [Thompson et al., 1994], but then often adjust that alignment by eye to create the final alignment.

Phase 3: Tree Inference

A phylogenetic reconstruction method, such as maximum parsimony, maximum likelihood, Bayesian analysis, or a distance-based method [Swofford et al., 1996, Felsenstein, 2003], is applied to the aligned sequences. Here we briefly describe maximum parsimony and maximum likelihood:

- **Maximum Parsimony (MP)** is an optimization problem based on the minimum evolution principal. A maximum parsimony tree is one that minimizes the number of changes over the edges of the tree needed to "explain" the sequences at its leaves. We formalize this goal here by first defining the parsimony score of a tree.

  Given a tree T that is leaf-labeled by a set of sequences S each of length l, an **extension** f of S on T is a labeling of all vertices of T by sequences of length l that maintains the original leaf-labeling by S. For two sequences x and y, let $H(x, y)$ be the Hamming distance between x and y. (For two sequences $x = x_1x_2...x_l$ and $y = y_1y_2...y_l$, $H(x, y) = |\{i : x_i \neq y_i\}|$.) Then the **parsimony score** of an extension f, denoted $score(f, T)$, is $\Sigma_{(u,v)\in E(T)}H(f(u), f(v))$. A **minimal extension** is one that minimizes the parsimony score, and the parsimony score of a minimal extension is the **parsimony length** of the tree T. Computing the parsimony length of a fixed leaf-labeled tree is $O(nrl)$, where n is the number of sequences, l is the length of the sequences in S, and r is the number of states observed in S [Fitch, 1971, Hartigan, 1972]. In the case of DNA, $r = 4$.

  A maximum parsimony tree for a given set of sequences S is a tree that has the smallest possible parsimony length of any tree leaf labeled by S. While computing the score of a single tree can be done in linear time, finding a minimum parsimony tree is equivalent to the Hamming-distance Steiner tree problem which is known to be NP-hard [Foulds and Graham, 1982]. An exhaustive search for a maximum parsimony tree on n sequences would require looking at all $(2n-5)\cdot(2n-7)\cdot...\cdot 1$ possible topologies of trees with the given leaf-labelings. Branch-and-bound algorithms can reduce the search space, but do not reduce the asymptotic running time. Using branch-and-bound algorithms researchers can compute all maximum parsimony trees for datasets of up to

about 25 taxa before the running time becomes prohibitive. Thus, for larger datasets, heuristic searches are performed whereby the tree-space is explored until a local optimum is found.

- **Maximum Likelihood (ML)** is also an optimization problem, and seeks the most probable tree according to some specified model of evolution. Given a model of sequence evolution, a maximum likelihood tree is a tree topology, along with a set of branch lengths and parameters for the given model of evolution, that maximize the conditional probability of observing the sequences at the leaves given the assumed model. Again let S be a set of sequences of equal length. The maximum likelihood solution for the set S is an edge-weighted tree (T, w), together with a set of model parameters $\mathcal{M}$, that minimizes $Pr(S|T, w, \mathcal{M})$. As with MP, ML is NP-hard [Chor and Tuller, 2005], and heuristic searches are used to compute ML trees.

Phase 4: Post-processing

In the post-processing stage, scientists summarize the results obtained in the tree inference stage. This typically involves computing a consensus tree for tree inference methods, such as MP, that return multiple trees (for details on efficient ways of storing and processing sets of trees see Boyer et al. [2005]), and assessing the support for branches in the inferred tree(s). In Chapter 4, we investigate the performance of a supertree method that makes use of bootstrap support values, a particular type of branch support information.

Here we briefly describe the non-parametric bootstrap technique; for a more in depth discussion, see Felsenstein [2003, pg. 335-363]. The goal of a non-parametric bootstrap analysis is to estimate the variability of a phylogenetic estimate produced by a particular reconstruction method on a given dataset. In practice, these analyses are used to estimate how well supported a particular phylogenetic estimate is

supported by the input data.

A non-parametric bootstrap analysis involves three steps. First, many pseudo-samples of the input data are generated. Each pseudo-sample, which we will call a **bootstrap replicate**, is obtained by sampling with replacement from the columns of the original alignment to obtain an alignment of the same length. Second, a given phylogenetic method is applied to each bootstrap replicate; each tree estimated on a bootstrap replicate is called a **bootstrap estimate**. Finally, the frequency of each bipartition observed in the bootstrap estimates is reported. For a given bipartition σ, the **bootstrap value** of σ is the proportion of bootstrap estimates that contain an edge that induces σ. The results of a bootstrap analysis are often presented by annotating the edges of phylogenies estimated on the original dataset with their bootstrap values.

2.4 Simulation Studies

Simulation studies are widely used to evaluate the performance of phylogenetic reconstruction methods, as well as that of supertree methods. Simulation studies are important for two reasons:

- When performing a phylogenetic analysis of real data, we do not typically know the true tree for the taxa under study. In simulation studies we know the true tree because we have explicitly constructed it, and can use this tree to measure the topological accuracy of the tree(s) returned by a given tree inference method.

- Furthermore, the theoretical guarantees on performance for phylogenetic methods are usually very loose. Simulations help overcome this problem by providing actual figures on the topological accuracy the trees returned by phylogenetic reconstruction methods, as long as the model space is adequately

explored, and the models are biologically relevant.

2.4.1 A traditional simulation study

A typical simulation study in phylogenetics involves the following four steps:

- **Step 1** Generate either a random or a biologically-based model tree on which sequence data will be evolved. A random model tree can be picked uniformly at random from all possible tree topologies or by some random process (a pure birth or a birth-death process, for example). A biologically-based model tree is one produced by a thorough phylogenetic analysis of a biological dataset, and thought by experts to be the most accurate estimate of the history of that set of taxa.

- **Step 2** Choose a stochastic model of sequence evolution and generate sequences under this model on the model tree generated in Step 1; this produces a set of aligned sequences at the leaves of the tree.

- **Step 3** Apply the reconstruction method(s) being evaluated to the aligned sequences, and apply appropriate post-processing procedures, discussed in the previous section, for the given method and estimated topologies.

- **Step 4** Compare the estimated tree(s) to the model tree topology using, for example, false positive and false negative rates.

The procedures in Step 1 and the model in Step 2 are often varied to evaluate how the method(s) in Step 3 will perform under different model conditions. For each model condition, i.e., for each way of executing Steps 1 and 2, many replicates are produced, and Steps 3 and 4 are applied to each replicate dataset for each model condition. The scores computed in Step 4 are then averaged over all replicates for each model condition, and possibly over all model conditions; standard deviation and standard error are also computed.

2.4.2 Supertree Simulation Studies

Supertree methods are also evaluated using simulation studies. Recall that for supertree methods, the input is not aligned sequences, but a set of source trees. Thus, in supertree simulation studies, one must choose how to generate the set of source trees to which the supertree methods under evaluation will be applied.

Because supertree methods do not deal with the sequence data directly, many construct tree topologies without applying a phylogenetic reconstruction method to sequence data, either by selecting subtrees (or slightly perturbed subtrees) of the model tree or generating random source trees [e.g., Lapointe and Levasseur, 2004, Holland et al., 2007]. Such simulations test supertree methods under conditions that are far from those encountered by biologists using supertree methods in practice, where the source trees are inferred from sequence data. Therefore, these types of simulation studies are less likely than biologically-based simulated data to inform how the supertree methods will perform in empirical use.

Supertree simulation studies that generate source trees from sequence data [e.g., Bininda-Emonds and Sanderson, 2001, Eulenstein et al., 2004, Criscuolo et al., 2006] follow a process similar to the simulation study outlined in the previous section, with a few additions to Step 2, and substituting supertree methods for the typical phylogenetic reconstruction method in step 3. After generating a large suite of sequence data in step 2, source trees are reconstructed on subsets of those data (subsets of the characters as well as taxa). In Step 3, the supertree method under investigation is applied to these sets of source trees.

I discuss supertree simulation studies further in Chapter 3, and in Chapters 4 and 5, I use simulation studies to evaluate the performance of various supertree methods including one of my own design, SuperFine.

2.5 Matrix Representation with Parsimony

The most popular supertree method is Matrix Representation with Parsimony (MRP), where the input trees are encoded as a matrix of partial binary characters which is then analyzed under parsimony. Under standard MRP, each of the source trees is represented by a matrix of 0s, 1s, and question marks (?s). For each edge in the source tree, we define one character with all the taxa on one side of the edge, assigned state 0, all the taxa on the other side assigned state 1, and any taxon that does not appear in the source tree coded as missing (and hence assigned a question mark). These data matrices are then combined into one matrix, the *matrix representation*, and a tree is computed on the resultant matrix using parsimony. Again, given that parsimony is an NP-hard problem, heuristic searches are used.

The following property of MRP is folklore, and the proof is omitted.

Theorem 2.5.1. *Let $\mathcal{T}$ be a compatible set of trees and let $MR(\mathcal{T})$ be the matrix representation of $\mathcal{T}$. Then a tree T has minimum parsimony length for $MR(\mathcal{T})$ if and only if T displays $\mathcal{T}$.*

Weighted MRP [Ronquist, 1996], is a variant of MRP. A weighted MRP analysis differs from MRP in that the source trees are presumed to have weights on the edges (which are defined by support values generally computed using bootstrapping). These trees are then represented by binary matrices with missing data as for MRP, but each column in the matrix is assigned the weight given to its edge. This weighted matrix is then analyzed using weighted parsimony. Weighted MRP was shown in Bininda-Emonds and Sanderson [2001] to outperform MRP.

In Chapter 4, I assess the accuracy of MRP and weighted MRP and compare their performance to the combined analysis approach. In Chapter 6, I discuss another variant of MRP, minimum fractional weighting, that is based on rooted triples.

2.6 Related Work

Supertree methods are typically evaluated in terms of the topological accuracy of the resultant phylogenies, using simulation studies. Previous simulation studies have investigated how different properties of the input and phylogenetic analysis impact the final phylogenetic accuracy [Bininda-Emonds and Sanderson, 2001, Chen et al., 2003, Burleigh et al., 2004, Eulenstein et al., 2004, Lapointe and Levasseur, 2004, Piaggio-Talice et al., 2004, Ross and Rodrigo, 2004, Chen et al., 2006, Criscuolo et al., 2006].

Nearly all supertree simulation studies have followed the methodology used by Bininda-Emonds and Sanderson [2001] [e.g., Chen et al., 2003, Bininda-Emonds, 2003, Eulenstein et al., 2004, Piaggio-Talice et al., 2004, Chen et al., 2006, Criscuolo et al., 2006]. I am interested in discussing the methods used to construct the source trees and combined datasets. The following description is, thus, limited to those features.

Bininda-Emonds and Sanderson used 8-32 taxon, non-ultrametric model trees generated using the Yule-c pure-birth process in the r8s program. They generated either two or ten genes, each containing 500 nucleotides. Each gene was generated under a K2P+Γ model with a transition/transversion ratio of 2.0 and site-to-site rate heterogeneity with shape parameter 0.5 on the same model tree using the Seq-Gen program. All genes were available, though possibly not used in a source tree, for all leaf taxa.

To select the taxon set for a given source tree, Bininda-Emonds and Sanderson selected some proportion of the full taxon set uniformly at random. The selection probability, and hence the proportion of the full data set included, was constant across all source trees generated for a given replicate, and was set to either 0.25, 0.5, 0.75, or 1.0. To construct a source tree for a given gene and selected set of taxa, Bininda-Emonds and Sanderson estimated a tree on the sequences of the selected

taxa for that gene using a PAUP* parsimony or parsimony bootstrap search. The combined dataset consisted of the full set of genes, but only included, for a given gene, those sequences from taxa that were included in the source tree for that gene.

The major limitation of this study was that each source tree contained taxa sampled randomly from the entire model tree. While, in empirical supertree studies, the comprehensiveness of taxon sampling in the in-groups varies depending on the purpose of the study, the resources available to the researchers, and the ability to collect or access source material, there is a clear non-random distribution of taxon-sampling effort in the individual trees that would be used as input either for a supertree method or for producing a supermatrix.

All simulation studies are limited in that they cannot explore all the complexities of biological processes and systematic practices which influence characteristics of the data that may in turn affect the performance of phylogenetic methods. The generality of the results of any given simulation study may not hold in a broader context. To my knowledge, all previous supertree simulation studies have either followed the methodology of Bininda-Emonds and Sanderson [2001], or used some less biologically realistic methods, to generate source trees. Given the limitations of the methodology used in previous studies, simulation methods that produce source trees in a more realistic way are needed. In Chapter 3, I present an alternative methodology that not only approximates the taxon-sampling efforts of systematists, but also simulates some biological factors that influence the genes and taxa included in source tree datasets, as well as many other features that allow the source trees to more closely approximate those used in empirical supertree studies.

Chapter 3

SMIDGen: a Supertree Method Assessment Platform

3.1 Introduction and Motivation

The simulation platform described here, which I have named SMIDGen (Super-Method Input Data Generator), goes beyond previous supertree simulation studies by expanding the number of biological factors used and by representing a broader range of the types of choices a systematist would make. Both of these are critical elements that impact the characteristics of the data generated. In the design of this platform, I explored many facets of how one can generate data for use in a supertree simulation study by considering the biological and systematic influences that come into play at each step of the data generation process for real data. Here, I list some of the most important issues that would ideally be considered when designing a supertree simulation study:

- Should the source trees be generated from sequence data generated on some model tree, or should the source tree topologies be predetermined?

- If model trees are used, what type of model trees should be used, and how many taxa should the study include?
- Since supertree methods are often used to construct a tree on multi-gene datasets, there are many considerations that involve generating many genes.
  - Should all the simulated genes evolve under the same model of sequence evolution?
  - Should they evolve on the same tree?
  - Should they evolve at the same rate?
  - Should gene gain and loss be incorporated into the gene data generation?
- Should taxonomic bias, the plethora of genetic data available for mammals vs. the dearth for nematodes, for example, be included when selecting source tree datasets?
- Since biologists choose the genes appropriate for studying a particular group based on evolutionary rates, should we include rate preferences in our gene selection process?
- How does the supertree approach compare to the combined analysis approach?

In this chapter, I pose more questions than I answer as I describe a simulation platform designed to help researchers answer many of the questions listed above, and I look to empirical supertree studies to determine what factors to include in this platform.

I wanted to build a platform on which to test the topological accuracy of supertree methods and the alternative approach, combined analysis. In particular, I wanted to compare the performance of supertree methods to that of combined analysis methods when analyzing datasets consisting of many genes, where the majority of the genes are not available for all the taxa, due to both biological and systematic

processes. In this dissertation, I use the term "missing data" to refer to the absence of gene sequences due to this limited availability of gene sequences. Among the biological processes that lead to missing data are gene "gain" via either duplication or horizontal gene transfer, and gene loss due, for example, to a gene becoming non-functional. I do not model any of these processes specifically in my simulation platform; I simply model a more generic gain and loss of genes over the model tree (this process is described in detail in Section 3.2.2). Systematic practices that lead to missing data include clade-focused studies, funding limitations, sampling bias, and sparse sampling. I include the systematic practices of clade focus and sparse sampling in larger groups via a process detailed in Section 3.2.3.

Section 3.2 presents the capabilities and options of SMIDGen and discusses why these features were included and what impact they might have on the datasets and reconstruction produced. Section 3.3 details how this simulation platform relates to previous simulation studies and to empirical supertree studies.

3.2 Software Capabilities, Options, and Processes

The supertree simulation software, SMIDGen, produces datasets appropriate for testing the performance of supertree and combine analyses approaches via the following four steps (the specific procedures and available options are detailed in Sections 3.2.1 through 3.2.4):

- **Generate a model tree.** Model trees can be generated using some stochastic processes, or by reading in a model tree provided by the user.

- **Generate gene sequences.** Evolve a number of genes on the model tree topology. Each gene is "gained" exactly once in the tree, and lost at others; once a gene is lost, it does not reappear. The gene gain nodes and loss nodes are selected by a random process that depends upon the model tree topology

and branch lengths.

- **Select source tree datasets.** Create data sets for phylogenetic analyses, differing in the taxa and genes used, so as to replicate the taxon-sampling strategies used by systematists. In particular, the user can generate both datasets that are focused on a specific clade of the model tree and datasets that include that are selected broadly from the entire model tree. The former are called *clade-based datasets*, while the latter are called *scaffold datasets*; the motivation for including both types of datasets is detailed in Section 3.2.3.

- **Construct source trees and a combined datasets.** The source tree datasets produced in Step 3, are then used to construct source trees, applying a tree inference method to each dataset, and to produce a combined dataset for use in a combined analysis.

In the remainder of this chapter, I refer to a single pass through the above process as a *replicate*, and to a set of such replicates for which all the option settings remain unchanged as a *batch*. The more replicates in a batch, the clearer picture one obtains of the datasets generated under the specified model conditions.

3.2.1 Model Trees

One can use biological trees or random trees (trees drawn from some distribution of tree topologies) as the model tree. A biological model tree is one which is reconstructed from a thorough analysis of a biological dataset, and believed by experts to be the most accurate estimate of the history of that set of taxa. The user can specify the number of replicates that use a given biological model tree. A single batch using a biological model tree consists of any number of replicates that use the same biological model tree.

The user could also choose to use random model trees, drawn from a biolog-

ically realistic distribution of tree topologies. SMIDGen currently uses r8s [Sanderson, 2003] to generate random model trees using either a pure birth or a birth-death process[Ross, 1996], but by making a simple change in the code it could use some other software or process.

The r8s program generates model trees in the following manner. First, it generates an ultrametric model tree under the process selected by the user, with the targeted numbers of taxa, and tree height of 1.0. It then modifies branch lengths to deviate from ultrametricity by applying a random scaling factor to each edge length in the tree. At the root, the scaling factor is set to 1.0 and progressively altered, parent branch to daughter branches, by adding a value drawn from a normal distribution with mean zero and standard deviation set by the user. The scaling factors are constrained to be at least 0.05 and no greater than eight (with a small code change the allowed range for the scaling factors can be adjusted).

Ultimately, the variables that the user control in the random model trees produced by r8s are the random process used, the number of taxa, and the deviation from ultrametricity (and potentially the range of scaling factors). The *average height* of a tree is the average weighted path distance from the root to leaves of the tree; all trees are generated with an expected average height of one. As I will discuss in Section 3.2.2, this software allows the user to adjust the effective height of the tree on a gene-by-gene basis by scaling the evolutionary rate of the genes individually.

A single batch using a particular combination of random process, number of taxa, and deviation from ultrametricity, contains many replicates each using a different tree produced with these specifications. So while biological model trees are used a batch contains replicates that use the same model tree, when random model trees are used, each replicate is performed on a different tree topology drawn from the desired distribution. (If the user wishes to perform many replicates with the same random model tree, they must simply run multiple single replicate batches

using the same random seed for the model tree generation.)

3.2.2 Gene Data Generation

In the gene data generation phase of our simulator, we incorporate several biological factors that can lead to missing data and other characteristics of the available sequence data for supertree analyses. The gene data generation is a two step process. First, we determine which taxa have a particular gene by modeling gene gain and loss over the model tree. The location of these gain and loss events in the tree determines the taxa in which that gene will be present, and this defines the subtree of the full model tree that will be used in the second step, gene sequence generation. One can adjust the distribution of genes over the tree by changing the gene gain and loss rates (the statistics for evaluating this distribution are discussed in Section 3.2.5).

The gene gain and gene loss rates are parameters that affect the location of the "birthplace" of a gene and subsequent losses; for a single replicate the gain and loss rates are fixed for all genes in that replicate. A gene can have only one birthplace on a given tree. To determine the birthplace of a gene, we evolve a single binary site down the tree, with the gene gain rate as the rate of evolution of that site.

Initially, this site takes state zero, representing the absence of the gene. Branch lengths are normalized so that the distribution is independent of tree height. On each branch of the tree the probability of a gene gain occurring is given by the probability $1 - e^{-\lambda b}$, where λ is the gene gain rate, and b is the normalized branch length. An outcome is randomly generated, with the above probability, along each branch leading away from a parent node, and if a gene gain occurs then the state of the parent node is set to 1, representing the presence of the gene. This parent node is the birthplace of the given gene. If a birth does not occur, a recursive call is

made to all children. If several children of the same node are returned, then we pick the earliest. If several children of the same node are returned with equal dates, pick one at random with equal probability. In addition to setting the gene gain rate, the user can choose to have a minimum and maximum theoretical gene-clade size. This simply limits the location of the birthplace of the genes, so that the number of taxa in the clade defined by that birthplace is in a certain range.

After the birthplace is chosen, another binary site is evolved down the tree using the same decay process starting at the birthplace with state 1, and with the gene loss rate as the rate of evolution. For each branch on which a change from 1 to 0 is observed, the descendants of that branch will not have the given gene: after the gene has been lost it cannot be regained in subsequent lineages. Thus, there can be several edges where the gene is lost but only one birthplace. If you take the set of taxa that contain a given gene, that set is pathwise connected on the model tree, and the most recent common ancestor of that set is the birthplace of the gene. This subtree constitutes the model tree topology for the given gene.

I will refer to genes whose presence and absence is determined by the process above as *non-universal* genes to distinguish them from *universal* genes which are born at the root of the tree and may or may not ever be lost. If a universal gene is not allowed to be lost as it evolves on the tree, then it is truly universal in that it are present in all taxa. The user can, however, choose to allow the universal genes to be subject to the gene loss process described above. The user can specify the number of non-universal and the number of universal genes that are used in a given replicate.

After determining the underlying tree for a given gene (universal or non-universal), the next step is to choose a rate at which the gene will evolve. We can evolve genes under three different rates of evolution, creating slow, medium, and fast genes. One way we can achieve these different rates of evolution is by multiplying

every branch of the tree by a single scaling factor, using a different factor for each desired rate of evolution. This changes the height of the model tree, scaling it by the given factor. For each of these classes of genes, we can assign a value by which the branch lengths of the model tree will be multiplied. The three evolutionary rates, together with the two options for universality, result in the possibility of six different types of genes. The user can specify how many of each of these six types will be created.

In addition, the user can, for an individual gene, scale the rate of evolution by a different value for each edge. Instead of selecting the scaling factor to be constant over the entire tree, we could, for each edge, select the scaling factor from a normal distribution with mean and variance determined by the class of the gene (slow, medium, or fast). (Other distributions could easily be incorporated should they be desired.) The expected height of the tree for a gene would then be the mean specified for that class of gene. For multi-gene datasets, this is similar to the No Common Mechanism (NCM) model [Steel and Penny, 2000], under which each individual site evolves under its own rate of evolution. While the rate distributions we offer are not as varied as those produced under a NCM model, they are more complicated than those commonly used in supertree simulation studies which assume that all genes evolve under rates drawn from a common distribution.

The model of sequence evolution for each gene is chosen randomly from a set of GTR+Γ+I parameters provided by the user. These parameters can be set for simpler models of evolution, such as Jukes-Cantor or K2P.

The sequence length is the same for all genes in a given replicate, and can be set to any positive integral value.

With these options we can generate genes under a variety of models of evolution. Within a single run, we can vary the relative rate of evolution between the genes and the model of sequence evolution of the genes. From run to run, we can

vary the sequence length and vary the birth and death rates of the genes, which eventually determines which taxa have a given gene.

3.2.3 Dataset Selection

After we produce a large suite of genes that constitute the biologically available data for a study, we select subsets of that data on which to construct source trees. The selection of source tree datasets was designed to simulate the processes that systematists might follow in various kinds of studies.

Systematists often concentrate on lower level taxonomic groups (genera, families, and sometimes orders), sampling heavily within the group of interest and much less comprehensively in the groups used as outgroups. We refer to these as *clade-based* studies.

In addition to these taxonomically low-level studies, *scaffold* phylogenies have been constructed for higher level taxonomic groups, e.g., angiosperms [Soltis et al., 1997] and metazoa [e.g. Glenner et al., 2004]. Scaffold phylogenies sample taxa widely distributed across the group to provide a broad-scale sense of the relationships of the lower-level groups contained within the higher-level group. In the context of a supertree analysis, scaffold phylogenies can provide the necessary "glue" for connecting phylogenies. In the absence of scaffold phylogenies, the only overlapping taxa between source trees will often be the small number of taxa used as outgroups for the clades of interest. Thus, supertree efforts will often consist of a large number of clade-based phylogenies that are densely sampled within their clades of interest and a small number of broadly distributed scaffold phylogenies based upon markers that evolve slowly (see Table 3.1). We implemented dataset selection for both types of datasets, clade-based and scaffold.

Group (Reference)	Method	Num. taxa	Num. source trees	Num. taxa in scaffold*
Primates [Purvis, 1995a]	Hierarchical MRP	203	112**	203 (100%) (taxonomy)
Carnivora [Bininda-Emonds et al., 1999]	Hierarchical MRP	271	177**	not given
Hologalegina [Wojciechowski et al., 2000]	MRP (with topological constraints)	571	22	52 (9.1%)
Pinus [Schwilk and Ackerly, 2001]	MRP	95	14	47 (49.5%)
Bacteria [Daubin et al., 2001]	wMRP	37	130-196	37 (100%)
Mammalia [Liu et al., 2001]	MRP (large:small source trees weighted 4:1)	90	430	37 (41.1%)
Procellariiformes [Kennedy and Page, 2002]	MRP	122	7	90 (73.7%)
Chiroptera [Jones et al., 2002]	Hierarchical MRP	916	105	not given
Poaceae [Salamin et al., 2002]	1) wMRP (normal, purvis, and irreversible)	403	55	not given
	2) wMRP (c.a. source trees)	61	8	61 (100%)
Lagomorpha [Stoner et al., 2003]	MRP (robust:nonrobust source trees weighted 2.81:1)	80	146	not given
Lipotyphla [Grenyer and Purvis, 2003]	(w)MRP (most source trees were MRP supertrees)	184	147 (7 final)	scaffold is a supertree of 6 taxa
Angiosperms [Davies et al., 2004]	wMRP	379	46	323 (85.2%), 224 (59.1%)
Marsupialia [Cardillo et al., 2004]	MRP (source trees with identical taxon sets were combined using supertree methods)	267	158	267(100%) (taxonomy)
Cetartiodactyla [Price et al., 2005]	MRP	290	201	290 (100%) (taxonomy)
Eutheria [Beck et al., 2006]	MRP (some source trees were MRP supertrees)	113	725 (109 supertrees)	115 (100%)
Carcharhiniformes: Sphyrnidae [Cavalcanti, 2007]	MRP (non-weighted, weighted, purvis, and irreversible)	8	5	8 (100%)
Mammalia [Bininda-Emonds et al., 2007]	MRP (combined previously published supertrees and some of their own.)	4510	>2500 (31 supertrees)	not given

Table 3.1: Selected empirical supertree studies. For each supertree study we give the supertree method(s) used, the number of taxa in the final supertree, the number of source trees, and the number of taxa in the scaffold dataset. If we could not determine which source trees served as scaffold trees, we instead give the number of taxa in the largest source tree. ** Indicates the number of publications from which source trees were drawn, not the actual number of source trees.

Selecting clade-based datasets

As systematists typically focus on monophyletic groups, the selection of clade-based datasets begins by selecting a random clade as our group of interest. Since this selection is from using simulated data, with no predefined genus or families from which to choose, we determine the clade of interest by evolving a single site down the model tree, and choosing the point of the first mutation as clade of interest's root. To select the clade of interest we use the same process by which we chose a birthplace for each gene — the gain rate for this process can be set separately from that of the gain rate for genes. The clade of interest is defined to be all the descendants of the node selected by this process. The user can choose to set a minimum and maximum number of taxa in the clade of interest. If the selected clade is outside this range, then we reselect. We ensure that each clade-based dataset is based on a different clade; this leads to a limitation on the number of clade-based datasets we can produce for a single model tree. If the user requests more clade-based datasets than are available, we simply create as many clade-based datasets as there are clades that satisfy the minimum and maximum requirements on the number of taxa.

Once an appropriately sized clade of interest is selected, the next step is to choose the set of genes that best represents this clade. The simulation software allows two different ways of selecting genes. The first method is to set the parameter g, the number of genes to be used, and select the first g genes from a list of all the genes ordered by the number of taxa in the clade of interest that contain them. The second gene selection method uses a minimum threshold for the proportion of taxa from the clade that must have the gene present; all genes that are present in at least the minimum proportion of taxa in the clade are selected. The advantage of the first method is that the user has complete control over the sequence length of the clade-based datasets. While under the second option, all genes that are in some sense "good enough" are included. The genes a systematist would choose in reality,

might use some combination of these two criterion.

Besides taxonomic coverage, biologists might also consider what kind of genes should be used (fast, average, slow, perhaps a mix), as well as whether to use universal-type genes. I have not implemented any gene selection criterion based on gene speed, but I do allow the user to specify whether or not universal genes can be used in the clade-based datasets. The user should be aware that if they choose to allow universal genes in clade-based datasets while not allowing gene loss in the universal genes, then the universal genes will be selected over most non-universal genes for all clade-based datasets.

After we have selected the set of genes that will be used in the dataset, we can determine which taxa will be included. This again can be done in different ways. We have implemented two options, taking either the *union* or the *intersection* of the taxa from the clade represented by the selected genes. If one uses the union option, then all the taxa in the clade that have any of the selected genes are included. If one uses the intersection, only the taxa in the clade that have *every* selected gene are included, resulting in a dataset with no missing data for each clade based dataset, since our criteria are based upon taxonomic coverage rather than rate of evolution. Thus, using the union option includes more taxa, which would likely lead to denser taxon sampling and, hence, potentially more accurate trees [Heath et al., 2008]. However, this option also can lead to a dataset with significant amounts of missing data, which could negatively impact accuracy [Heath et al., 2008].

Selecting scaffold datasets

To apply a supertree method in a meaningful way, there must be sufficient coverage and overlap between the source trees. In practice and in the simulator, scaffold datasets serve as links between clade-based datasets that might not otherwise overlap with any other source tree dataset. The inclusion of scaffold datasets, depending on

how they are selected, does not guarantee the necessary overlap of the source tree datasets but does make it more likely. As we discuss in subsection 3.2.5, we provide some statistics about the overlap between source tree datasets, however, we chose to leave the check of a set of source tree datasets as a valid supertree input to the user in post processing.

The use of some sort of scaffold tree is common practice in empirical supertree studies. When possible the scaffold dataset is formed using some gene, or set of genes, found in taxa throughout the various groups in the full study [e.g., Kennedy and Page, 2002, Davies et al., 2004]. When such a dataset is not available, researchers often include a taxonomy to produce a dense and, often, unresolved scaffold tree [e.g., Purvis, 1995a, Cardillo et al., 2004, Price et al., 2005]. We have incorporated the former type of scaffold dataset into our simulation platform.

We approach gene and taxon selection differently for the scaffold and clade-based datasets. The scaffold dataset includes taxa throughout the full taxon set, so that it likely includes some taxa from each clade dataset. Each taxon in the model tree is included in the scaffold dataset with a probability, f, a parameter set by the user. We call this probability the *scaffold factor*. The expected number of taxa included in the scaffold dataset is fn, where n the number of taxa in the model tree. By adjusting the scaffold factor, one can investigate the effects of the density of the scaffold dataset. Indeed, in empirical supertree studies, we find a wide range of scaffold densities in their scaffold datasets (see Table 3.1), from a very sparse scaffold containing only nine percent of the full leafset [Wojciechowski et al., 2000], to scaffolds containing all taxa in the full leafset.

When constructing the clade-based datasets, we select genes based on their coverage of the clade of interest. For the scaffold datasets, there is no clade of interest. Instead, there is a set of taxa that have been chosen to be in the scaffold dataset. Therefore, gene selection is slightly different for the scaffold datasets than

for the clade-based datasets. If gene loss was not allowed in the generation of the universal genes, all universal genes are present in every taxon. In this case, taxonomic coverage is not a factor, and the user can only choose between universal genes based on their rate of evolution. If universal gene loss is allowed, however, taxonomic coverage of the genes is an issue and, in addition to rate of evolution, the same selection methods available for clade-based datasets can be used. If the selected genes do not include all taxa, one can choose either the union or the intersection options defined for taxon inclusion described above for the clade-based dataset selection.

The final factor that determines what data will be used to reconstruct source trees is the minimum number of taxa that can be included in a source tree. If, after the dataset selection processes described above, a given source tree dataset, clade-based or scaffold, has less than this minimum, the dataset is simply not included in the study. This can be used to avoid computational problems that arise using trees of three or fewer taxa, or simply to ensure that the datasets used are in some desired range. The clade-based datasets together with the scaffold dataset(s) make up the set of source tree datasets.

3.2.4 Constructing Input for Supertree and Supermatrix Analyses

Once we have selected the source tree datasets, both clade-based and scaffold, we construct a tree on the union of the leafsets of the source tree datasets. We offer several different types of methods for reconstructing source trees, including distance-based, parsimony, and maximum likelihood methods (see the Appendix for details on commands for methods described in this section). The user can choose among the following implementations of these methods:

- PAUP*'s implementation of Neighbor Joining. Logdet distances are used by default, but any of the methods available in PAUP* can be used through our

software.

- PAUP*'s maximum parsimony and maximum likelihood based search techniques.

- A PAUP*-based parsimony ratchet. (Since this implementation is used in Chapter 4, I give the details of this method here.) Starting trees are generated from a random addition sequence, and followed by one round of TBR swapping. Once a local optimum is reached, a ratchet iteration is performed. The first step of the ratchet iteration randomly reweights 25% of the characters with weight two, while keeping the weight of the other characters 1.0. A round of TBR hill-climbing is then performed on the reweighted data matrix. During the second step, the weights on all characters are returned to 1.0, and another round of TBR hill-climbing is performed.

- RAxML maximum likelihood search. I provide the default RAxML maximum likelihood search which uses GTRCAT, as well as the GTRMIX option.

- Parsimony-based bootstrap analysis, implemented using PAUP*. (Again I provide the details here, since this implementation is used in Chapter 4.) This search uses the "fast MP" command (`bootstrap nreps=1000 search=faststep`) in PAUP* to analyze 1000 bootstrap replicates, and returns the majority consensus of these analyses annotated with bootstrap values.

- Likelihood-based bootstrap method implemented using RAxML.

We now consider the following question: if one wants to compare supertree methods to combined analysis, what data should one include in the combined dataset: only the data used directly to compute supertrees, or all available data? If one uses the first, more conservative, option then one is comparing what each method can do based on the same initial data. However, one may be debating between using

supertree methods and combined analysis as a way to analyze a large, incomplete, multi-gene dataset. In this case the focus is to determine how best to take advantage of, or put to use, all available data. Here, the second, more inclusive, option is appropriate. We implement both options for constructing the combined dataset, a conservative method and more inclusive method. The conservative method only includes the data from the source tree datasets, that is, the exact gene–taxon combinations that are represented in some source tree dataset. The inclusive method assumes that we were restricted to the taxa and genes in the source tree datasets, but that we have access to all biologically available genetic information for those genes and those taxa, not just the specific taxon-gene combinations in the source tree datasets. In particular, if a given gene is included in some source tree dataset, then for every taxon in any source tree, if that gene is present in that taxon, that sequence is included in the combined dataset.

3.2.5 Statistics and Accuracy Measures

SMIDGen reports several statistics about the genes generated and datasets and trees constructed over each run which can be used to evaluate the impact of the various settings described in this section. Here we describe the statistics that are reported for every run and batch of simulations, and we mention which of the settings described above may effect these statistics.

The software also reports the distribution of genes among the taxa in two ways. First we provide a histogram of the number of taxa that have a particular number of genes present. Second, for each gene, we provide (1) the number of taxa in the clade defined by the birthplace of that gene, (2) the number of taxa in that clade that retained the gene, (3) the proportion of taxa in that clade that retained the gene, and (4) the number of loss events in the gene's evolution over that clade. For these four values, we also provide the average over all universal genes and over

all non-universal genes.

SMIDGen also reports the number of taxa in each scaffold and clade-based dataset. Since each taxon is included in the scaffold with a probability equal to the chosen scaffold factor, the number of taxa in the scaffold dataset may vary. If gene loss in the universal genes has been turned off, the number of taxa in the scaffold dataset follows a binomial distribution where the mean is the product of the scaffold factor and the number of taxa in the full model tree. If universal gene loss is allowed, then the number of taxa in the scaffold dataset also depends on which chosen taxa contain the selected genes and whether the gene-taxa-intersection or the gene-taxa-union option is used to determine which taxa will be included in the scaffold dataset. For each clade-based dataset, we report the number of taxa in the clade of interest, the number of taxa that are eventually chosen to be in the dataset for that clade, and the proportion of clade members that are chosen to be in that dataset. The latter of these two values depends on which genes are chosen for that dataset, which of the taxa contain the selected genes, and whether the clade-based datasets include the intersection or the union of taxa that contain the selected genes. We also report the number of taxa and the sequence length for each source tree dataset as well as for the combined analyses.

We report a few statistics about the overlap between the source tree datasets. We give the average, maximum, and minimum overlap between each pair of clade-based datasets, as well as the average and minimum overlap of the scaffold with each clade-based dataset. These values are impacted by the gene birth and death rates, the clade birth rate, and the source tree selection settings.

3.3 Discussion of Possible Applications

SMIDGen can assist in answering many questions about the performance of supertree methods. I now have a simulation platform for testing supertree methods

where the processes that lead to the final collection of source tree datasets has been generated in a way that approximates the biological and systematic factors that can influence them. Additionally, biologists can now explore the effects of these different factors because they can be set by the user. For example, 1) one could test the effect of the distribution of rates among the genes, 2) one could test methods of source tree reconstruction and supermatrix methods that can use partitions of the data, where each partition evolves on the same tree but under different models of sequence evolution, or 3) one could test the effect of different numbers of source trees and varying sizes of source trees. I used SMIDGen to test the effects of the number of taxa in the model tree, the density of and the number of genes used in the scaffold dataset, and the reconstruction methods used to generate source trees and supertrees. I present my findings in the next chapter.

Finally, researchers doing empirical supertree studies can use SMIDGen to approximate the datasets they have and to test different supertree methods to see which would be best for their situation. If they are ambitious, they could even add the exact methods they plan to use (or have used) for source tree generation and supertree construction and see how this compares to alternatives. SMIDGen could, with some small changes, replicate several previous supertree simulation studies.

Chapter 4

Comparison of Existing Supertree Methods and Combined Analysis

In this chapter, I am interested in comparing the performance of current supertree methods on datasets that reflect both biological factors and the practices of systematists that effect datasets used in empirical supertree studies. In the previous chapter, the goal was to incorporate into SMIDGen many of the biological and systematic factors that influence the characteristics of biological data available for a supertree study. In this chapter, I choose a few of these factors to explore, and set the remaining parameters to best reflect how supertrees are used in practice.

The primary objective in this study was to determine the relative performance of current supertree methods and combined analysis, each based upon either MP or ML, under more realistic conditions than have previously been considered. In particular, I aimed to determine how the design of a systematics study, in terms of selection of markers, subsets of taxa, and phylogenetic estimation method, impacted these issues. I explored the effect on the final supertree of the properties of the scaf-

fold dataset, as well as the number of taxa in the overall data set. As the scaffold is the glue that holds the rest of the trees together, its quality and characteristics are likely have a significant impact. No supertree simulation studies to date have addressed the influence of characteristics of the scaffold datasets on supertree accuracy. Similarly, nearly all existing supertree simulation studies include less than 100 taxa, while most recent empirical supertree studies significantly exceed 100 taxa, some going into the thousands. Therefore, the need is great for simulation studies with large numbers of taxa in the full dataset, modeling the effect of increased numbers of taxa on the tree reconstruction methods. I, therefore, chose to explore the effects of the number of taxa in the full dataset, the density of the scaffold, the number of genes used to construct the scaffold, and the phylogenetic method used to construct trees, both source trees and combined analysis, from sequence data.

In summary, the experimental study presented in this chapter addresses the following issues:

1. How does the degree of completeness of the taxon sampling in the scaffold tree affect the accuracy of the supertree?

2. How does the accuracy of the reconstruction of the scaffold tree and the other source trees affect the accuracy of the supertree?

3. Given reconstruction of input trees on only the taxa of the input trees, do supertree methods that use a measure of support for internal branches of the input trees, such as weighted matrix representation parsimony (wMRP), perform better than those that do not, e.g., MRP?

4. How does the phylogeny estimation procedure impact the relative performance of supertree vs. combined analysis approaches?

5. Does the relative performance of combined analysis and supertree methods change as the number of taxa increase to levels seen in large empirical supertree

studies?

4.1 Methods

I overview the experimental methodology here, and provide details of the methodology in the remainder of this section.

Step 1: Random model trees, containing 100, 500 or 1000 taxa, were generated using a Yule pure birth process [Ross, 1996].

Step 2: On each model topology, the evolution of a number of different genes was simulated such that each gene evolved independently of the other genes according to a GTR+Γ+I model, under gene-specific parameter values. All genes were "gained" exactly once in the tree, and were lost at others; once a gene was lost, it did not reappear. The birthplace and gene loss nodes were selected by a random process that depended upon the model tree topology and branch lengths. Universal genes were present at every node in the tree. Each gene had a sequence length of 500.

Step 3: For each model tree topology, data sets for phylogenetic analyses were created, differing in the taxa and genes used, replicating the taxon-sampling strategies used by systematists. The result of this experimental process was a collection of data matrices which were then used for combined and supertree analyses.

Step 4: The source tree datasets were combined for use in the combined analyses, and on each source tree datasets, ML and MP source trees, appropriate for use in weighted and unweighted MRP, were computed.

Step 5: Phylogenies on each data set were estimated using four different supertree methods (weighted and unweighted versions of MRP, each based upon source

trees estimated in turn by either maximum parsimony or maximum likelihood), and two combined analysis approaches (one based upon maximum parsimony and one based upon maximum likelihood).

Step 6: Error rates of the reconstruction methods were computed by comparing the estimated trees to the true trees with respect to the unrooted topologies, and recorded the running time of each method.

Steps 1 through 4 were performed using the simulation platform presented in Chapter 3. I, therefore, use the options and terminology defined in that chapter to detail the procedure for each of those steps. In Steps 1-3, I varied the total *number of taxa*, the *scaffold factor*, and the *number of scaffold genes*, which I will refer to as n, f, and g, respectively. I will refer to a given setting of n, f, and g, as a model condition.

4.1.1 Step 1: Generate Model Trees

I used SMIDGen to generate model trees having 100, 500, and 1000 taxa, using the pure-birth option for the tree generation process, and set the deviation from ultrametricity to 0.5. For the model conditions with $n = 100$ and $n = 500$, I set the number of trees generated 30, and to 10 for the model conditions with $n = 1000$. Recall that, when using random model trees the simulation software produces a single replicate for each model tree. Thus, this study used 30 replicates for each 100- and 500-taxon model condition and 10 for each 1000-taxon model condition. The smaller number of replicates for the 1000-taxon model conditions was due to the very long running times for the 1000 taxon analyses.

4.1.2 Step 2: Evolve Gene Sequences

For each replicate model tree, SMIDGen then generated a large suite of genes available for use in inferring the source trees. 100 non-universal genes, and 5 universal

genes were produced.

Gain and loss patterns.

The loss rate for the universal genes was set to zero, so that each was present at all taxa in the model tree. The gain rate for the non-universal genes was set to 0.3, and the loss rate to 0.1.

Rate classes

Of the 100 non-universal genes, 25, 50 and 25 were assigned to be in the slow, medium, and fast rate classes respectively. All five universal genes were assigned to be slow genes – reflecting the practice in systematics to use slower evolving genes for higher taxonomic groups. I chose to scale each gene tree by a single value for their given rate class (rather than scaling each branch by a rate drawn from a normal distribution for the rate class). The slow, medium, and fast rates were set to 0.1, 1.0, and 2.0, respectively.

GTR+Γ+I parameters

The GTR+Γ+I parameters were chosen with equal probability from a pool of parameter sets estimated by Ganapathy [2006] on three biological datasets:

- Angiosperm data set – 288 aligned DNA sequences of a group of Angiosperms, each of length 4811 [Soltis et al., 1997, Brauer et al., 2002].
- Nematode data set – 682 aligned small subunit rRNA sequences, consisting of 678 species of Nematodes and four outgroups, each of length 1808 [Baldwin et al., 2008].
- rbcL data set – 500 aligned rbcL DNA sequences each of length 1398 [Chase et al., 1993].

Data Set	Substitution Matrix			Base Frequencies.	Prop. Invar. Sites	Gamma
Angiosperm	1.54755	3.67531	1.86115	A = 0.223269	0.2	0.5
		0.93047	4.53303	C = 0.206748		
			1.0	G = 0.256568		
				T = 0.313414		
Nematode	1.24284	3.47484	0.48667	A = 0.300414	0.273196	0.362026
		1.07118	4.38510	C = 0.191363		
			1.0	G = 0.196748		
				T = 0.311475		
rbcL	1.09397	3.12811	0.35141	A = 0.320128	0.101878	0.397524
		1.55972	3.64704	C = 0.176726		
			1.0	G = 0.167462		
				T = 0.335683		

Table 4.1: Model parameters estimated on three biological data sets and used for generating DNA sequences.

In Ganapathy [2006], the GTR+Γ+I parameters for each dataset were estimated by running a PAUP* ML heuristic search for 60 days on a fixed tree. The PAUP* command used to obtain these estimates was *lscores /base=estimate nst=6 rmat=est rates=gamma ncat=10 shape=estimate pinvar=estimate lcollapse=no*. The tree for the angiosperm dataset, obtained by running GARLI [Zwickl, 2006] from a random starting tree. At the time, this tree had the best known ML score for that dataset. The trees for both the nematode and rbcL datasets were generated by applying RAxML, in its default GTRCAT setting, to a PAUP-Ratchet starting tree. Table 4.1 presents the evolutionary model parameters estimated on each dataset.

4.1.3 Step 3: Dataset Production

For each model tree, SMIDGen created DNA sequence data sets for phylogenetic analyses by selecting subsets of the suite of gene data generated on that model tree. To mimic taxon-sampling strategies used by systematists, data sets differed, both in the taxa and genes used, and in whether they were scaffold or clade-based. The result of this process was a collection of data matrices, which were then used to generate input for the combined and supertree analyses.

Selecting clade-based datasets

For each clade-based data set, a clade of interest was selected from the model tree using a similar process to that used in finding a birth node for each non-universal gene (see Section 3.2.3 for details); the decay rate for this process was set to 0.25. The size (number of taxa) of the clade of interest was restricted by setting bounds on the number of extant taxa in a clade to avoid selection of either very small or very large clades. For each 100-taxon model tree, five clades were selected with a clade size of at least 20. For each 500-taxon model tree, 15 clades were selected with a clade size of at least 30, and for each 1000-taxon model tree, 25 clades were selected ranging in size between 30 and 500.

The decision to keep the number of clade studies small was an algorithmic one. SMIDGen requires that each clade study be based on a different clade of interest. Given the settings chosen for the minimum clade size, there were a limited number of possible clades of interest for a particular model tree. If one tried to generate more clade studies than there were possible clades of interest, the code would have only returned only as many unique clade studies as existed for a given replicate. This may have resulted in some replicates having different numbers of clade-based datasets than others. To keep such differences from effecting the evaluation of the effects of the model conditions under study, the number of clade studies was limited.

For each clade chosen, the three non-universal genes were selected that covered the largest number of taxa in the clade, breaking ties randomly. Once these three non-universal genes were chosen, the taxa in the clade were restricted to only those that had all three of the genes. That is, I used the intersection option in final taxa selection for the clade-based datasets. This process produced clade-based datasets without any missing sequence data, but which may not contain all the taxa of the specified clade. This process can produce datasets with small numbers of

taxa, and I chose to exclude any clade-based data set that had fewer than ten taxa.

Scaffold dataset

The parameters used to generate the scaffold dataset varied between model conditions. In this study the scaffold factor f was set to either 0.20, 0.5, 0.75 or 1.0, and the number of scaffold genes g to either one, two or four. Only universal genes were allowed for use in the scaffold dataset. In some replicates for model conditions with smaller scaffold factors, I detected a handful of datasets with such low taxon overlap that it would have been inappropriate to apply either a supertree or a supermatrix analysis (see [Page, 2004, pg. 257] for a description of the conditions needed to apply a supertree analysis). These replicates were discarded from the study.

Because of the combination of scaffold and clade-based source trees, and because all were larger than some minimum size, the source tree datasets as a group had good coverage of most taxa in the model tree. However, the design choices in this study did cause the number of source trees to be smaller than the number of source trees for some biological supertree analyses in the literature (Table 3.1), which often have many very small source trees.

4.1.4 Step 4: Source Tree and Combined Dataset Construction

Combined dataset

The combined dataset was constructed from the source tree datasets using the more conservative method described in Section 3.2.4. That is, for each gene used in some source tree dataset, only those sequences contained in some source tree dataset were included in the combined dataset.

Source trees

For each source tree dataset, I inferred phylogenies using both MP and ML methods to construct source trees for both unweighted and weighted MRP analyses. The later supertree method requires that the source trees are annotated with bootstrap values. This meant constructing four sets of source trees: MP, MP bootstrap, ML and ML bootstrap. The MP source trees for the MRP analyses were estimated using the parsimony ratchet described in Section 3.2.4. The scaffold data sets were analyzed using 50 ratchet iterations, and the clade-based data sets using 100 iterations, keeping the most parsimonious solutions obtained over all the iterations. The MP source trees were the strict consensus of the most parsimonious trees found. The MP bootstrap source trees were estimated using a PAUP* to perform a bootstrap analysis as described in Section 3.2.4. The MP bootstrap source trees were the majority consensus of this analyses, with internal nodes of the tree labeled with bootstrap values.

To generate the ML input-trees for MRP analysis, I used RAxML [Stamatakis, 2006] in its GTRMIX default setting, returning the single best tree. The ML bootstrap source trees for weighted MRP were estimated using the fast bootstrap version of RAxML, analyzing 100 bootstrap replicates in its GTRMIX setting. The smaller number of bootstrap replicates for the ML analyses was necessary due to longer runtimes per replicate.

4.1.5 Step 5: Combined Analysis and Supertree Reconstruction

Combined analyses

For the combined analyses, phylogenies were estimated on the supermatrix created from the source-tree matrices using both maximum parsimony (CA-MP) and maximum likelihood (CA-ML). MP analyses consisted of five iterations of the parsimony ratchet, using the same implementation described for the source tree reconstructions

(again, returning the majority consensus of the most parsimonious trees found in the search). I chose to use five iterations of the parsimony ratchet in the CA-MP analyses due to time constraints.

One might argue that this does not give a fair comparison of the performance of CA-MP in comparison to other methods that were given significantly more iterations. In a test on the 100-taxon datasets, however, I determined that performing more iterations did not increase topological accuracy.

When 100 iterations were used, the resulting tree were no more accurate, with respect to the FN rate, than the tree obtained by running only 5 iterations (see Figure 4.1). Neither was any topological accuracy gained by running more ratchet iterations when accuracy is measured with FP (figure not shown). The parsimony length of the datasets on the trees resulting from the 100-iteration implementation were shorter than those resulting from the 5-iteration version (see Figure 4.2). This means trees with lower parsimony scores were not necessarily more topologically accurate, indicating that parsimony is not the correct optimization criterion for these datasets. This is not surprising, as the data were generated under a fairly complex model of evolution. A biologist using parsimony to analyze a real dataset would not, however, have the true tree available to choose between different methods based on topological accuracy, and would likely choose the method that returned trees with the lowest parsimony score. Therefore, in reality a biologist would likely perform the combined analysis using 100, or possibly more, ratchet iterations. Thus, due to time constraints, the CA-MP analysis deviates from what a biologist would realistically do. This small test indicates that the MP combined analysis in this study is perhaps more topologically accurate than a more realistic, more time consuming analysis would produce.

ML analyses used RAxML in its GTRMIX default setting for the 100- and 500-taxon data sets and in its GTRCAT default setting for the 1000-taxon data sets.

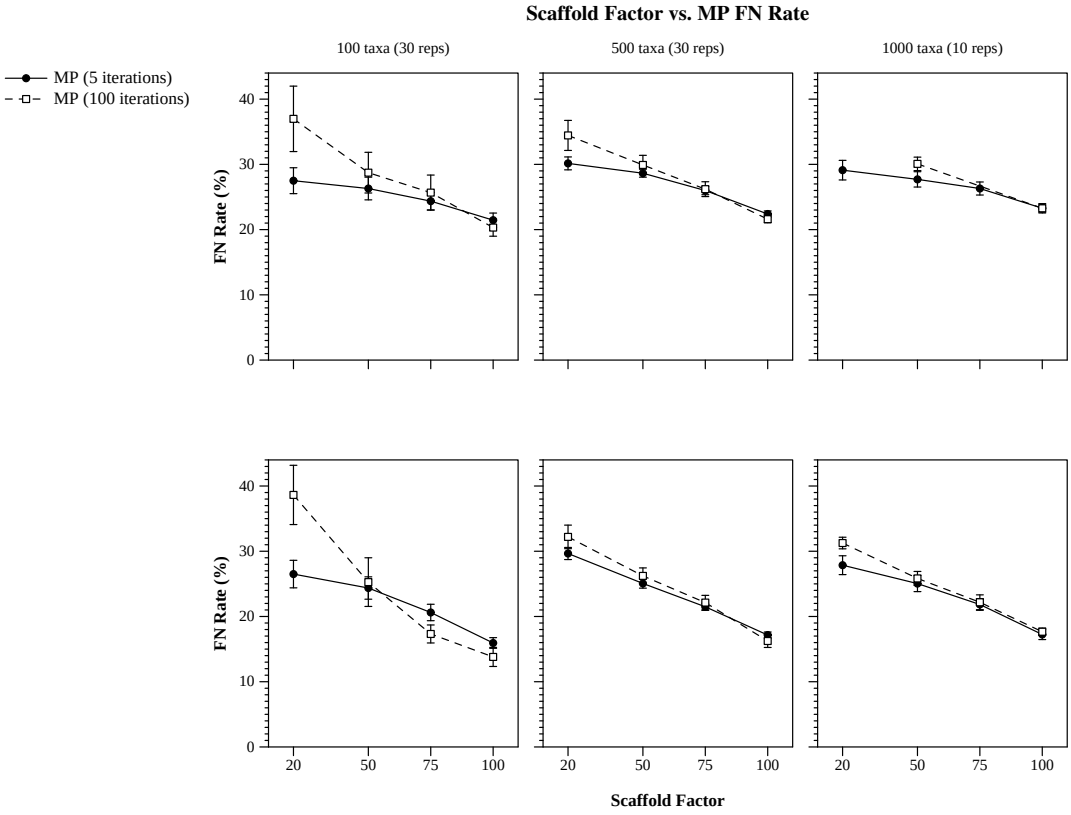

Figure 4.1: False negative (FN) rates (means with standard error bars) for MP combined analyses as a function of the scaffold factor. Graphs a-c are for data sets with one scaffold gene, and d-f for four scaffold genes. Values in italics on the x-axis are the average percent of missing data in the data matrices of the combined data sets for that scaffold factor.

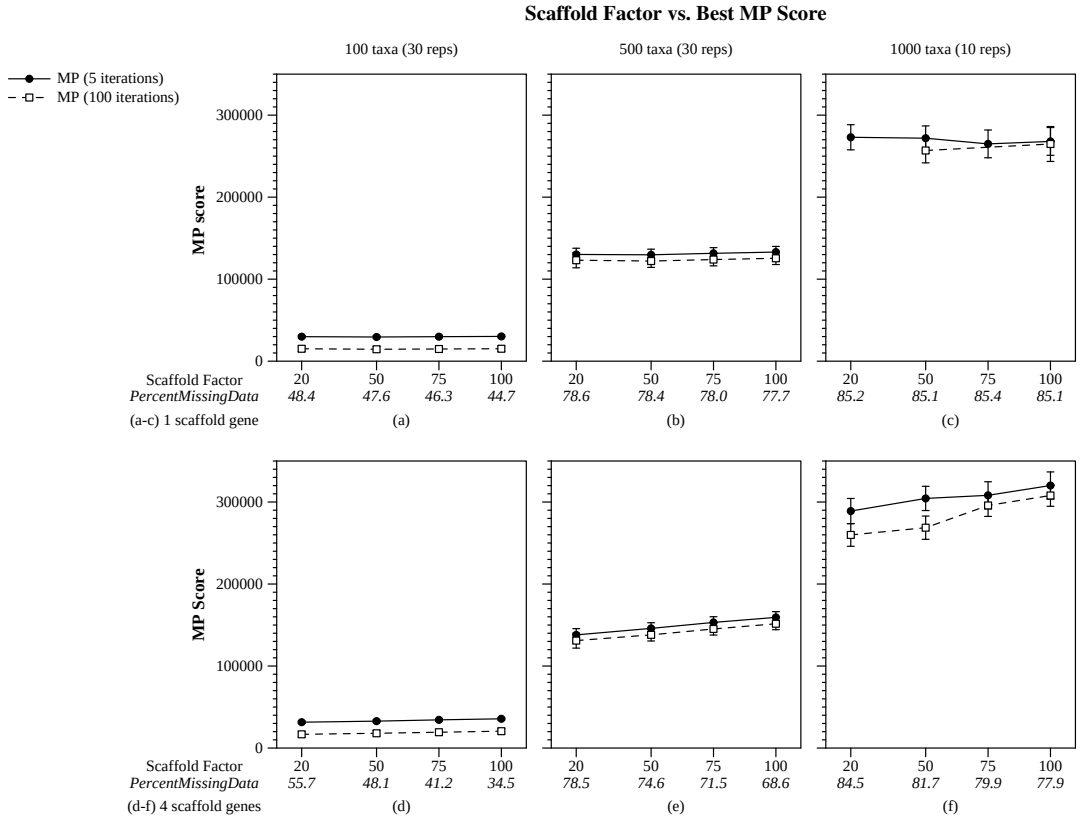

Figure 4.2: Parsimony scores (means with standard error bars) for MP combined analysis as a function of the scaffold factor. Graphs a-c are for data sets with one scaffold gene, and d-f for four scaffold genes. Values in italics on the x-axis are the average percent of missing data in the data matrices of the combined data sets for that scaffold factor.

GTRCAT was used for the largest data sets due to the prohibitive running time of GTRMIX. GTRMIX uses the same search algorithm as GTRCAT, but computes the likelihood score on the final tree, using the GTRGAMMA method. Thus, both methods produce the same trees, topologically.

Supertree Reconstruction

For the supertree analyses, the MP and ML source trees were used as input for the MRP method, and the bootstrap MP and bootstrap ML source trees as input to the weighted MRP method (wMRP); thus, MRP applied to MP source trees (MRP-MP) and wMRP applied to MP source trees (wMRP-MP) had slightly different source trees, and MRP applied to ML source trees (MRP-ML) and wMRP applied to ML source trees (wMRP-ML) also had slightly different source trees. For the wMRP analyses, the bootstrap proportions from the input-tree analyses were used as the branch lengths. Thus, there were a total of four different supertree reconstructions (MRP-MP, MRP-ML, wMRP-MP, and wMRP-ML).

I used r8s [Sanderson, 2003] to produce the matrices for the MRP and wMRP analyses from the given source trees. For the MRP analyses, supertree matrices were analyzed using 50 iterations of the parsimony ratchet described above. Since the PAUP*-based parsimony ratchet script used does not accept weights for branches, to perform the wMRP analyses I used a weighted parsimony search, with 100 random sequence additions, TBR branch swapping, and maxtrees=1000. For both MRP and wMRP analyses, I used the majority consensus of the most parsimonious supertrees returned by the search. The majority, instead of strict, consensus is used because it produces trees with a lower false negative rate (Fig. 4.3).

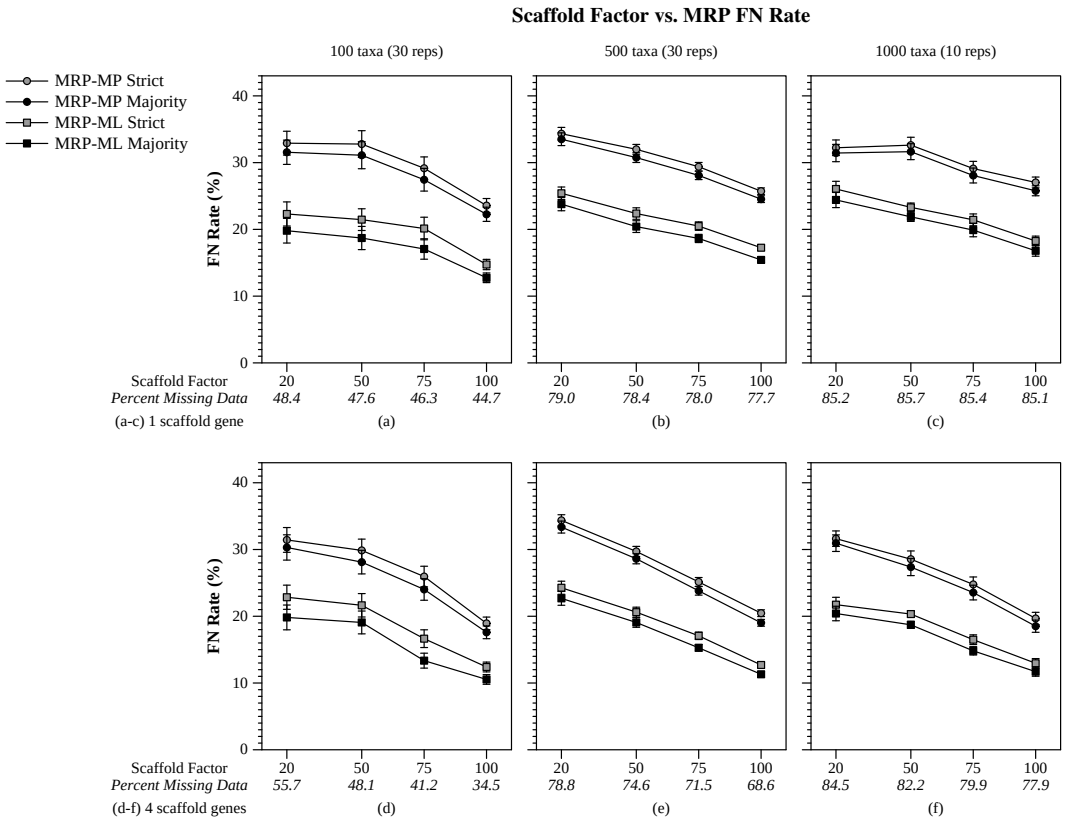

Figure 4.3: FN rates (means with standard error bars) for strict and majority consensus MRP supertrees as a function of the scaffold factor. Graphs a-c are for data sets with one scaffold gene, and d-f for four scaffold genes. Values in italics on the x-axis are the average percent of missing data in the data matrices of the combined data sets for that scaffold factor.

4.1.6 Step 6: Performance Evaluation

I computed the topological error rates of the reconstructed supertrees and combined analysis trees by comparing them to the true trees with respect to the unrooted topologies. Each calculation is restricted to the set of taxa in the estimated tree since not all estimated trees contained all the taxa in the full data set. The topological error rate was calculated using the false negative (FN) rate (see Section 2.1.1 for a description of this measure). I provide the average FN rates and standard deviations for each model condition.

I recorded the running time of each method on each dataset, and report the average over the replicates for each method under each model condition. However, wMRP-ML was not run on the 500 or 1000-taxon datasets, due to its excessive computational expense. Because the analyses were run under Condor (a distributed software environment [Thain et al., 2005]), the running times, for the larger datasets, especially, are inexact and larger than they would be if run on a dedicated processor; these are provided to give an approximate estimation of the time needed to perform these analyses.

Finally, this study explored the impact of the topological error of the source trees, the scaffold factor, the number of scaffold genes, and the number of taxa on the topological error of the resultant supertrees. In all cases, I report the average error rates calculated across the set of replicates for each of the specified datasets.

4.2 Results and Discussion

The six methods in this study had the same relative performance, with respect to topological accuracy, under all model conditions. CA-ML performed best, followed by wMRP-ML, MRP-ML, CA-MP, MRP-MP, and finally wMRP-MP (Figs. 4.4, 4.5, and 4.6). This result contradicts the conclusions of Bininda-Emonds and Sanderson

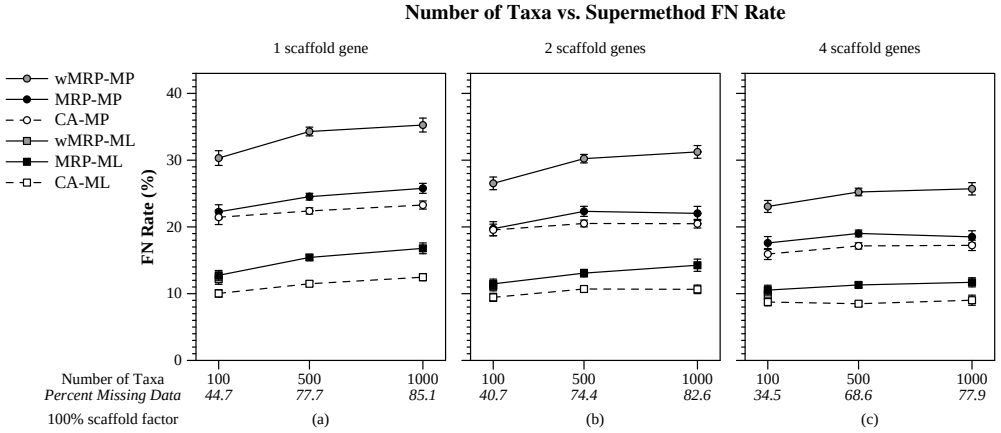

Figure 4.4: FN rates (mean with standard error bars) for supertree and super-matrix reconstructions as a function of the number of taxa in the model tree. Only data sets with 100% scaffold factors are presented.

[2001], and is likely due to several differences between the experimental designs, which I address in Section 4.3.

The parameter that had the biggest impact on the performance of the six methods studied here was the choice of optimization problem, i.e., whether maximum likelihood or maximum parsimony were used, and the second most significant parameter was whether a combined or a supertree analysis was performed. For example, there was a bigger difference in performance between CA-ML and CA-MP than between CA-ML, MRP-ML, and wMRP-ML (Figs. 4.5, and 4.6). Thus, the methods based upon ML gave the lowest error, and the methods based upon MP gave the highest error. While this is certainly not surprising, given the relative performance between ML and MP that has been observed in other simulations, it is still noteworthy for the following reason.

The sequence datasets analyzed using RAxML, for both the supertree and combined analysis methods, were obtained by concatenating gene data sets. Since each gene could evolve under a different model of evolution (all were GTR+Γ+ I, but had different parameter values), the data sets on which combined analyses and

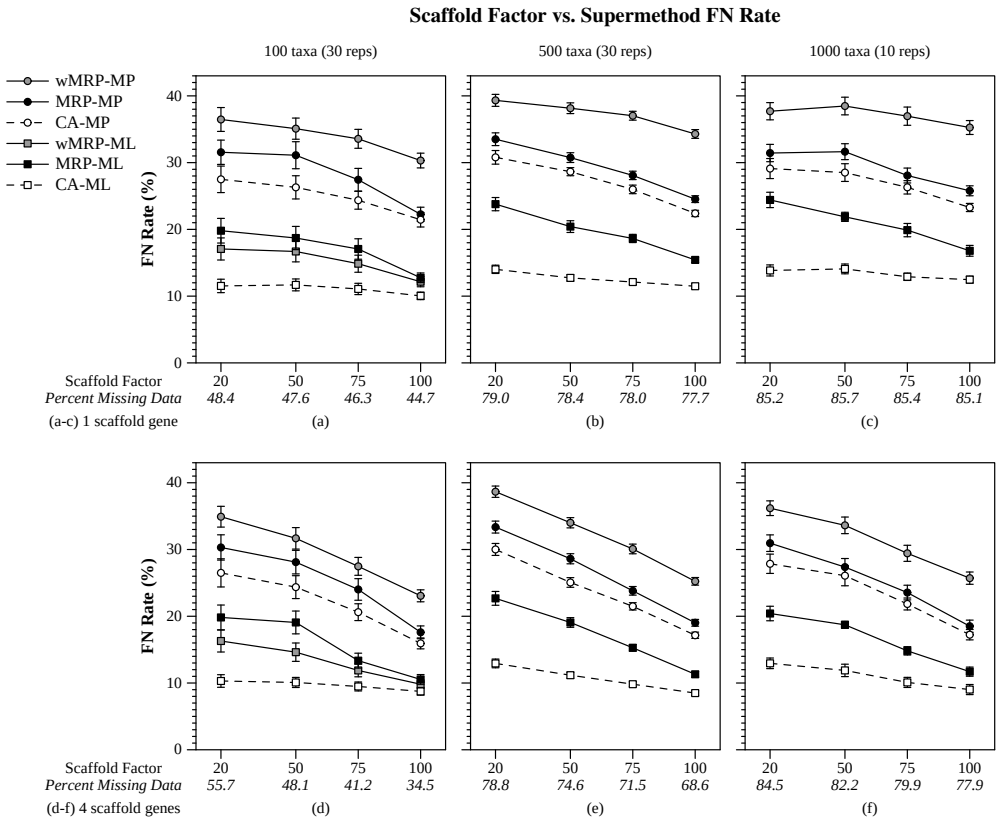

Figure 4.5: FN rates (means with standard error bars) for supertree and supermatrix reconstructions as a function of the scaffold factor. Graphs a-c are for data sets with one scaffold gene, and d-f for four scaffold genes. Values in italics on the x-axis are the average percent of missing data in the data matrices of the combined data sets for that scaffold factor.

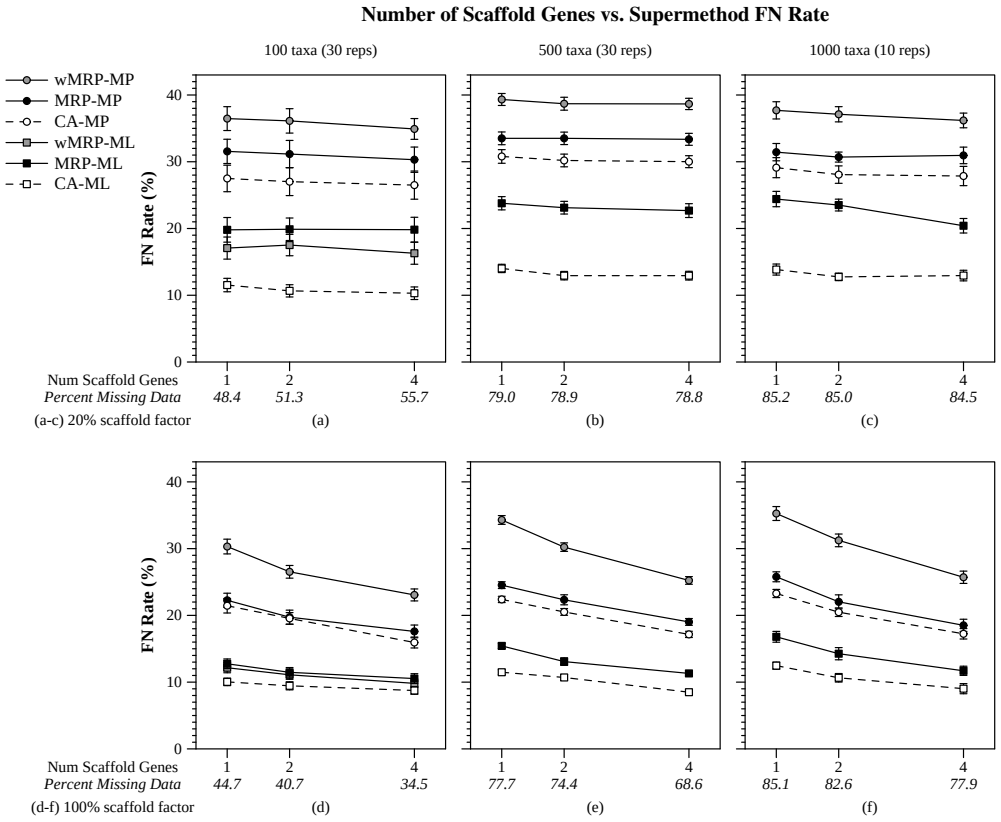

Figure 4.6: FN rates (means with standard error bars) for supertree and supermatrix reconstructions as a function of the number of scaffold genes used in the scaffold reconstruction. Graphs a-c and d-f are for data sets with 20% and 100% scaffold factors, respectively. Values in italics on the x-axis are the average percent of missing data in the combined data sets for that number of scaffold genes.

the majority of our source trees were based evolved under a mixed model. These mixed model datastes violate the assumption in RAxML that all the sites evolved under the same model. (RAxML can do partitioned analyses, but I did not utilize this option.) This could have made the ML-estimated source trees less accurate than if they were estimated under the correct partitioned models. That ML still outperformed MP shows that even without partitioning the data by gene maximum likelihood was still able to recover a more accurate topology than MP.

The advantage obtained by MRP and wMRP methods when using ML-based source trees instead of MP-based source trees was also substantial, but the explanation for the differences is more subtle. As expected, source trees estimated by ML were more accurate than the source trees estimated by MP (Fig. 4.7). Also as expected, both MRP and wMRP were sensitive to error in the estimated source tree, with supertree error increasing with increased source tree error. Supertrees estimated using ML source trees tended to have higher error than their average source tree, while supertrees estimated using MP source trees had the same or slightly lower FN rate than their average source tree. Thus it seems that when given more accurate input, ML trees, MRP and wMRP compound the error, and given source trees with more error, MP trees, these two methods still return supertrees with moderate error rates.

The accuracy of the MRP and wMRP methods also depended closely on the accuracy of the scaffold tree (Fig. 4.8). This was particularly evident for scaffold trees with 100% scaffold factor. The tighter relationship of supertree accuracy to scaffold accuracy rather than the accuracy of the clade-based trees was not surprising since the number of taxa in the 100% scaffold trees was much greater than the number in clade-based trees, and source trees with more taxa tend to more heavily influence the topology of trees produced by MRP [Purvis, 1995b]. I also found that supertree error rates were often lower than the scaffold tree error rates. A possible

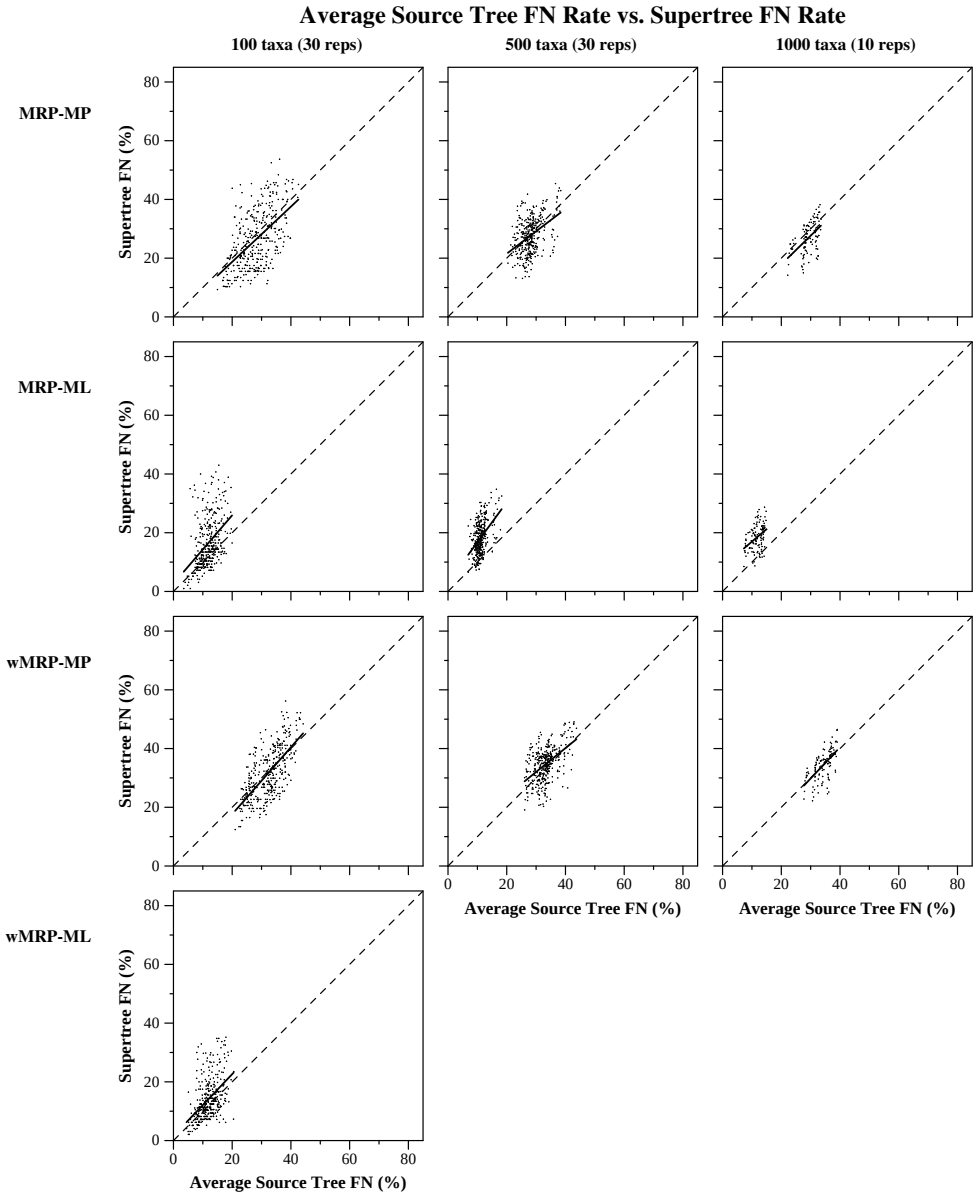

Figure 4.7: Average source tree FN rates against Supertree FN rates. Each point represents a single replicate for a single model condition. The solid line is a regression line. The dotted line represents supertree constructions that have the same FN rate as the average source tree given as input. Points above the dotted line correspond to supertrees that are less topologically accurate than the average source tree, while points below the line correspond to supertrees that improved upon the average accuracy of the source trees.

explanation is that scaffold trees tend to have higher error than clade-based source trees, probably as a result of having, on average, fewer sites, because scaffold trees are based upon one, two, or four genes, while clade-based source trees are based upon three genes.

While the relative performance between methods was consistent across all model conditions, certain experimental settings clearly impacted the differences between methods. In particular, the scaffold factor and number of taxa both impacted the differences between methods, with differences smaller when the data sets had only 100 taxa as opposed to 500 or 1000 taxa (Fig. 4.4), and when the dataset contained a very dense scaffold (Fig. 4.5). Increased numbers of scaffold genes improved tree accuracy, but a greater improvement was obtained when the scaffolds had high density (Fig. 4.6). These differences in performance were likely caused by how each method responded to the amount of missing data, since datasets that included scaffold genes that covered all the taxa (100% scaffold density) had a smaller proportion of missing data in comparison to datasets that had sparse scaffold genes (e.g., ones with 20% scaffold density) (Fig. 4.5). Therefore, increasing the number of scaffold genes that had low density increased the overall proportion of missing data, while increasing the number of scaffold genes that had high density decreased the proportion of missing data. Thus, scaffold density and number of scaffold genes interact to form a pattern of missing data that can impact the phylogeny reconstruction method.

In these experiments, I explored the trade-off between additional data and amounts of missing data by modifying the number of scaffold genes used to define the scaffold data set, and focusing on two scaffold factors: 20% (where adding scaffold genes adds substantial missing data) and 100% (where adding scaffold genes adds no missing) (Fig. 4.6). Additional sites improved the accuracy of all methods when the scaffold had no missing data, but only CA-ML and CA-MP reliably improved

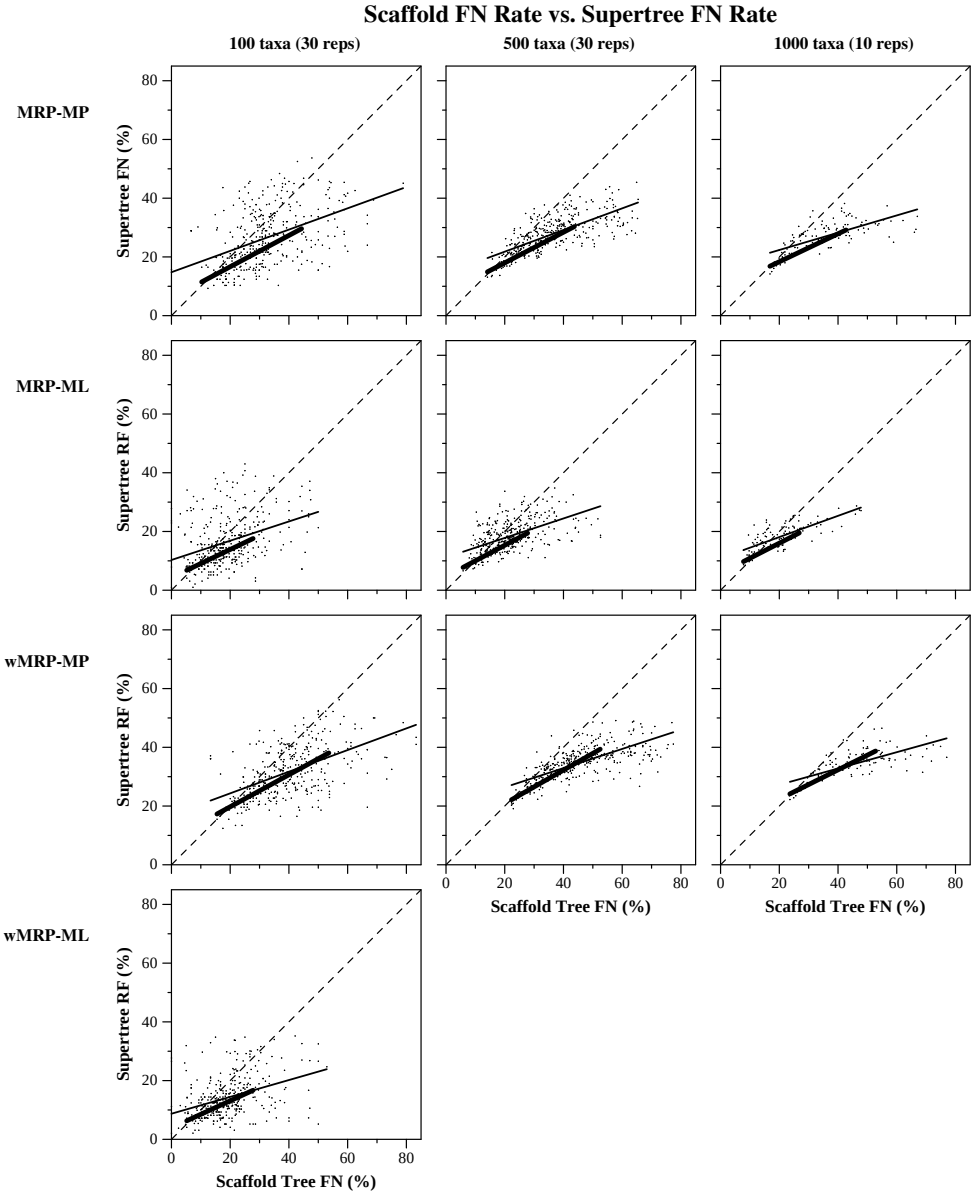

Figure 4.8: Scaffold tree FN rates against Supertree FN rates. Each point represents a single replicate for a single model condition. Thin lines are regressions for all data sets for the given method and number of taxa, and thick lines are regressions for data sets with a scaffold factor of 100%. The dotted line represents supertree constructions that have the same FN rate as the average scaffold tree included in the input. Points above the dotted line correspond to supertrees that are less topologically accurate than the scaffold tree, while points below the line correspond to supertrees that improved upon the accuracy of the scaffold tree.

when the scaffold had substantial missing data. This result suggests that weighted and unweighted MRP have a greater problem with substantial missing data than combined analysis. When scaffold factors of 20% and 100% were compared, the accuracy of CA-ML did not change much (Fig. 4.5), but all four types of MRP and CA-MP analyses improved substantially. This result supports the hypothesis that MRP is very sensitive to missing data, but combined analyses less so.

4.2.1 Running Times

Table 4.2 provides information on the running times of the methods studied here. Since these methods were run under Condor (and thus under a distributed system, and not on dedicated processors), these numbers should be considered approximate, and are given only as an indication of the general trends.

On the 100 taxon datasets, all methods completed in just a few minutes, with the exception of wMRP-ML, which took between seven and 17 hours. On these datasets, with the exception of observing wMRP-ML being extremely slow, no other comparison with respect to running time on these datasets is justified. However, the limiting factor in running wMRP-ML is the bootstrap analyses to produce the source trees, so that if trees with bootstrap support values are provided, the running time to apply wMRP would be at most ten minutes on these 100-taxon data sets.

On the 500 taxon datasets, MRP-MP, MRP-ML, and CA-MP all finished in under an hour, and were thus comparable in running time. These are therefore very fast methods. CA-ML and wMRP-MP were clearly slower, each using between four and 12 hours to complete (but roughly comparable to each other). I attempted to run wMRP-ML on these datasets, but each dataset took substantially more than a day to run, some about a week, so the analyses with wMRP-ML was not completed.

On the 1000 taxon datasets, MRP-MP, MRP-ML, and CA-MP took several hours (generally less than six hours for the MRP analyses, and between five and

Num. Taxa	Scaffold Factor	Num. Scaff. Genes	MRP		wMRP		Comb. Analysis	
			MP	ML	MP	ML	MP	ML
		1	0:02:26 (0:00:16)	0:04:12 (0:00:17)	0:01:59 (0:00:30)	7:00:43 (0:07:22)	0:00:19	0:09:22
	20%	2	0:02:46 (0:00:18)	0:04:33 (0:00:17)	0:01:59 (0:00:29)	13:09:51 (0:08:19)	0:00:20	0:14:49
		4	0:02:46 (0:00:18)	0:04:48 (0:00:17)	0:02:02 (0:00:30)	13:12:03 (0:09:27)	0:00:22	0:24:19
		1	0:02:31 (0:00:17)	0:04:52 (0:00:17)	0:02:02 (0:00:34)	13:27:05 (0:07:26)	0:00:20	0:19:28
	50%	2	0:02:52 (0:00:17)	0:04:43 (0:00:17)	0:02:00 (0:00:29)	13:34:02 (0:08:30)	0:00:19	0:18:45
100		4	0:02:36 (0:00:19)	0:04:47 (0:00:17)	0:02:03 (0:00:27)	13:29:32 (0:09:39)	0:00:21	0:11:47
		1	0:02:34 (0:00:18)	0:04:40 (0:00:18)	0:02:10 (0:00:38)	13:37:31 (0:09:14)	0:00:20	0:11:22
	75%	2	0:02:37 (0:00:21)	0:05:08 (0:00:19)	0:02:08 (0:00:31)	7:51:22 (0:04:47)	0:00:20	0:14:09
		4	0:02:39 (0:00:19)	0:05:25 (0:00:19)	0:02:12 (0:00:27)	16:08:55 (0:06:31)	0:00:22	0:17:46
		1	0:03:12 (0:00:19)	0:04:49 (0:00:18)	0:02:15 (0:00:40)	16:04:16 (0:00:46)	0:00:19	0:15:59
	100%	2	0:02:42 (0:00:19)	0:05:12 (0:00:18)	0:02:15 (0:00:31)	16:23:16 (0:00:32)	0:00:20	0:15:59
		4	0:02:51 (0:00:19)	0:05:24 (0:00:17)	0:02:27 (0:00:27)	14:12:23 (0:00:30)	0:00:21	0:16:42
		1	0:38:08 (0:14:00)	0:42:23 (0:10:30)	4:33:43 (4:11:26)		0:31:32	8:18:55
	20%	2	0:36:41 (0:13:12)	0:54:00 (0:14:02)	4:16:10 (3:52:54)		0:30:03	8:11:45
		4	0:36:15 (0:12:38)	0:43:26 (0:10:26)	3:40:18 (3:16:18)		0:31:41	10:46:11
		1	0:35:50 (0:12:57)	0:50:02 (0:11:11)	5:15:58 (4:54:09)		0:36:43	10:28:04
	50%	2	1:04:27 (0:23:12)	0:53:11 (0:12:19)	5:16:34 (4:50:07)		0:33:17	11:11:52
500		4	0:46:12 (0:15:36)	0:59:05 (0:13:42)	5:12:08 (4:42:06)		0:31:30	8:40:21
		1	0:38:20 (0:13:45)	0:54:45 (0:12:56)	7:03:10 (6:40:10)		0:37:44	7:02:24
	75%	2	0:37:54 (0:12:30)	0:52:09 (0:11:05)	6:34:58 (6:07:25)		0:34:19	7:53:34
		4	0:41:21 (0:12:39)	0:57:47 (0:11:43)	3:52:45 (3:18:24)		0:29:39	8:38:06
		1	0:44:33 (0:15:07)	0:57:14 (0:11:53)	7:04:20 (6:43:18)		0:35:42	7:51:45
	100%	2	0:44:38 (0:14:23)	1:09:41 (0:14:36)	12:33:24 (11:56:14)		0:37:26	7:59:34
		4	0:43:44 (0:12:02)	1:05:59 (0:11:38)	5:33:52 (4:55:00)		0:26:27	7:10:35
		1	3:27:39 (2:21:39)	3:14:56 (1:47:23)	85:59:18 (84:53:20)		6:09:44	30:51:13
	20%	2	3:22:14 (2:18:39)	3:26:56 (1:47:54)	30:34:14 (29:53:19)		5:46:56	28:05:57
		4	4:23:55 (3:05:25)	3:21:37 (1:51:27)	83:56:34 (82:42:34)		6:15:50	27:58:00
		1	3:22:33 (2:18:18)	4:22:00 (2:18:23)	75:21:00 (74:11:14)		7:36:14	33:21:22
	50%	2	3:36:43 (2:24:07)	3:28:32 (1:37:26)	54:19:41 (53:30:45)		6:14:55	29:23:03
1000		4	3:15:25 (2:01:12)	4:06:33 (2:00:27)	41:23:37 (40:19:38)		11:13:18	29:28:29
		1	4:12:25 (2:38:59)	4:18:28 (1:58:46)	88:13:11 (87:10:11)		8:07:53	33:39:42
	75%	2	4:01:05 (2:24:18)	4:22:16 (2:00:57)	67:17:02 (66:12:47)		6:16:32	28:41:11
		4	4:11:41 (2:22:43)	4:34:12 (1:40:56)	50:49:52 (49:25:52)		5:33:09	29:08:11
		1	5:29:37 (3:26:03)	4:29:46 (1:52:53)	174:47:23 (173:41:33)		7:56:46	34:14:48
	100%	2	4:26:05 (2:39:56)	5:53:10 (2:30:23)	295:10:35 (293:51:27)		6:02:16	27:11:45
		4	4:54:31 (2:43:09)	4:43:11 (1:51:43)	174:52:35 (172:49:10)		4:49:07	24:33:39

Table 4.2: Average running time for each of the six methods. Running times are given in hours:minutes:seconds. For the four supertree methods, the time to compute just the supertree is given in parentheses following the full running time, which includes the time taken to compute source trees.

11 for the CA-MP analyses). CA-ML took somewhat more than a day of analysis, and thus was significantly slower than the MRP and CA-MP analyses. In contrast, wMRP-MP was extremely slow, requiring more than two days on many datasets and some taking a week or more. The reason wMRP-MP was so slow on these datasets is the parsimony ratchet could not be used, and thus a standard weighted parsimony search was used, which takes a long time if run reasonably.

4.3 Comparison with Earlier Studies

Although my study reveals trends that are largely consistent with both Bininda-Emonds and Sanderson [2001] and Criscuolo et al. [2006] with respect to the relative performance of MRP and combined analysis, it provides a *distinctly different* conclusion than Bininda-Emonds and Sanderson [2001] with respect to wMRP and combined analysis. The differences between my results and those of Bininda-Emonds and Sanderson [2001] could be due to several factors, of which I consider the following to be the most likely: (1) the number of taxa in the model trees used in each study, (2) the methods used to select taxa for the source tree data sets. The former issue is probably the most significant in terms of the impact on the relative performance of combined analyses and supertree studies; however, the later issue may also turn out to be important.

The numbers of taxa employed in the two studies were quite different. While I examined datasets with at least 100 taxa and up to 1000 taxa, Bininda-Emonds and Sanderson [2001] explored datasets with only eight to 32 taxa. In my study, the accuracy of MRP-MP and wMRP-MP is almost identical on 100 taxon datasets, with the gap in performance increasing with the number of taxa; thus, wMRP-MP is much less accurate than unweighted MRP-MP on the larger datasets. These observations support the possibility that weighted MRP-MP might be better than unweighted MRP-MP on small datasets. Since most recent empirical supertree

studies have included upwards of 200 taxa (Table 3.1) and it is likely that future empirical supertree analyses will also include hundreds or even thousands of taxa, my results may be a better indicator of the relative performance of MRP-MP, weighted MRP-MP and combined analyses for current uses of supertree and combined analysis methods.

Selection of taxa for the source trees is also handled quite differently in the two studies, but it is not clear how much impact it has on the outcome. For my experiments, I produced two types of source datasets: scaffold datasets, and clade-based datasets. The scaffold datasets were a random sample of the taxa, but the clade-based datasets were obtained by first identifying particular clades, and then picking the genes that covered those clades best. Thus, my datasets included two very different ways of sampling taxa for source datasets, based upon systematic practice. The technique used by Bininda-Emonds and Sanderson [2001] only produced scaffold datasets. That is, their taxa were always randomly selected from the full dataset. The comprehensiveness of taxon sampling in in-groups in biological studies does vary, depending on the purpose of the study, the resources available to the researchers, and the ability to collect or access source material. There is, however, almost always a clear non-random distribution of taxon-sampling effort in most of the individual trees that would be used as input for a supertree method or for producing a supermatrix. Thus, while it is not clear whether the use of clade-based trees would affect the relative performance of the methods studied, it is clear that their technique for producing the source datasets is not as consistent with systematic practice as is the one used here.

4.4 Conclusions

I have several major observations. First, the choice of base method, MP or ML, has a greater impact on the phylogenetic accuracy of the final tree than whether a

supertree or a combined analysis is performed, with ML-based methods substantially outperforming MP-based methods. Second, when applying a supertree method, the error of the source trees is compounded in the resulting supertree (Fig. 4.7). Third, the density of the scaffold dataset (Fig. 4.5) impacts the accuracy of all six methods, but especially the accuracy of supertree methods. And finally, in almost all cases, combined analyses significantly outperform supertree methods, with supertree methods and combined analyses having comparable accuracy only for relatively small datasets which include a highly accurate and very dense scaffold (covering all or almost all the taxa), a condition that is rarely met. These results indicate that neither MRP nor wMRP are likely to be competitive with combined analyses in most realistic conditions facing systematists.

Thus, for researchers considering whether to use a supertree or combined analysis approach, my results indicate that combined analysis performed using a maximum likelihood search would produce a more accurate tree, particularly if there is no single data set that includes most of the taxa of interest. Additionally, while CA-ML is not the fastest method investigated here, with current software such as RAxML its running time is by no means prohibitive and the improved accuracy is worth the wait. For those who do not have the option of performing a combined analysis, my results provide guidelines for the design of a supertree study that may help minimize error in the supertree. Because increasing the density of the scaffold improves accuracy, biologists seeking to perform a supertree analysis should procure the most complete study possible for inclusion in their analyses. In addition, attention should be placed on obtaining the most accurate source trees possible, since source tree error tends to be compounded in the supertree analysis. Thus, while many types of data and methods have been used to produce source trees [e.g. Liu et al., 2001, Kennedy and Page, 2002, Salamin et al., 2002, Stoner et al., 2003, Grenyer and Purvis, 2003, Price et al., 2005, Bininda-Emonds et al.,

2007], systematists should be conservative about these decisions. If the researcher has access to ML bootstrap source trees, or has access to the sequence data and the time to perform ML bootstrap analyses, weighted MRP should be used for computing the supertree.

Chapter 5

SuperFine: A New Supertree Method

The previous chapter demonstrated a clear need for better supertree methods, particularly when handling datasets with sparse scaffold trees. I have developed such a supertree method. I will show that my method, which I call SuperFine, produces more accurate supertrees than MRP on datasets with sparse scaffold trees, while being as accurate as MRP on datasets with dense or complete scaffolds. Thus, SuperFine is preferable to MRP under conditions where the application of supertree methods is most attractive. Furthermore, in my experiments, SuperFine runs in less time than MRP. That is, SuperFine is more accurate than MRP when analyzing overlapping datasets for which no single dataset contains the majority of taxa under investigation. In addition, these analyses show that SuperFine is almost as accurate as combined analysis and is as robust as combined analysis to increasing amounts of missing data.

Section 5.1 presents two preliminary studies that motivated the development of the SuperFine method. In Section 5.2, I describe the SuperFine procedure and some of its theoretical properties. In Section 5.3, I describe a simulation study

designed to evaluate the performance of SuperFine. In Section 5.4, I present the results of this simulation study, which show that SuperFine outperforms MRP on the datasets with sparse scaffolds and is comparable to MRP under all other model conditions tested. Finally, in Section 5.5, I discuss possible ways to improve the performance of SuperFine, as well as the utility of this method beyond constructing supertrees.

5.1 Motivation and Preliminary Studies

With the goal of developing a new, more accurate supertree method, I investigated the use of two existing methods in new ways. Quartet Max Cut (QMC) [Snir and Rao, 2008] is a quartet-based phylogenetic reconstruction method. I investigated QMC as a supertree method by applying it to various encodings of the source trees as sets of quartet trees. As I will show in Section 5.1.1, our investigations revealed that the accuracy of the supertrees produced by QMC increase as the number of quartet trees used in the encoding of the source trees increases. However, for all but the densest quartet encodings tested, QMC returns a supertree that is less accurate than that returned by MRP. For the 100-taxon datasets, applying QMC to the set of all induced quartet trees of the source trees produces supertrees that are more accurate than MRP. Unfortunately, the memory requirements of this encoding means that it cannot be used to construct supertrees on datasets with 500 taxa or more.

I was simultaneously investigating the use of Strict Consensus Merger (SCM) [Huson et al., 1999a, Roshan et al., 2004a] as a general supertree method. SCM is a method that constructs a source tree from a set of source trees by merging pairs of trees until only a single tree is left. I tested the performance of SCM by applying it to various orderings of the pair-wise mergers. I found that one could easily obtain supertrees with low false positive rates, no matter what ordering method was used.

However, none of the ordering methods explored here produced SCM trees that had a lower false negative rate than that of MRP. I present the findings of this investigation in Section 5.1.2.

Alone, neither QMC nor SCM was a scaleable supertree method that produced accurate supertrees. Their relative strengths and weaknesses, however, led me to combine the two approaches and, ultimately, to the creation of SuperFine. As I will show in Section 5.4, SuperFine produces highly topologically accurate supertrees and is scaleable to large datasets.

5.1.1 QMC Applied to Various Quartet Encodings

QMC takes as input a set of quartet trees Q and produces a tree with leaf set $L(Q) = \bigcup_{q \in Q} L(q)$. Given such a set Q, finding the tree T that maximizes the number of elements of Q displayed by T is a computationally difficult problem [Steel, 1992]. QMC is a heuristic that tries to optimize this criterion. I tested the performance of QMC applied to various types of quartet encodings of input trees. For each source tree T I define the following quartet encodings:

- **All.** Select $q(T)$, all $\binom{|L(T)|}{4}$ induced quartet trees of T.
- **Topological Short.** For each edge $(u, v) \in E^{\circ}(T)$, let $t_1, ..., t_i$ be the i subtrees rooted at vertices adjacent to u that do not contain the vertex v, and let $s_1, ..., s_j$ be the j subtrees rooted at vertices adjacent to v that do contain the vertex u. For each $a \in \{1, ..., i\}$, let l_{t_a} be the leaf of t_a that has the shortest path distance to u, breaking ties arbitrarily; for each $a \in \{1, ..., j\}$, let l_{s_a} be the leaf of s_a that has the shortest path distance to v, breaking ties arbitrarily. Let

$$Q_{toposhort}(T, (u, v)) = \{l_{t_a} l_{t_b} | l_{s_c} l_{s_d} \in q(T) : a \neq b, \text{and } c \neq d\}.$$

Then $Q_{toposhort} = \bigcup_{e \in E^\circ(T)} Q_{toposhort}(T, e)$

- **k-Short.** When using the k-short selection method it is assumed the T is an edge-weighted tree where the weight of a given edge represents the evolutionary distance between the taxa at either endpoint; if T has no edge weights, each edge is assigned unit weight.

  For each edge $(u, v) \in E^\circ(T)$, and for each $i \in \{1, ..., 4\}$ let $t_1, ..., t_i$ and $s_1, ..., s_j$ be defined as above (in the description of topological short). For each $a \in \{1, ..., i\}$, let L_{t_a} be the k leaves of t_a with the shortest weighted path distance to u, breaking ties arbitrarily; for each $a \in \{1, ..., j\}$, let L_{s_a} be the k leaves of s_a with the shortest weighted path distance to v, breaking ties arbitrarily. If t_i contains less than k leaves then $L_i = L(t_i)$. Then let

  $$Q_{k\text{-}short}(T, (u, v)) = \{wx|yz \in q(T) : \quad w \in L(t_a),\ x \in L(t_b),\ y \in L(s_c),\ \text{and}$$
  $$z \in L(s_d) \text{ such that } a \neq b,\ \text{and } c \neq d\}.$$

  Then $Q_{k\text{-}short} = \bigcup_{e \in E^\circ(T)} Q_{k\text{-}short}(T, e)$.

- **Geo.** In the geo encoding, induced quartet trees are included with a probability inversely proportional to the cube of their topological diameter. The diameter of a quartet $q \in q(T)$, denoted $diam(q, T)$, is the maximum pair-wise path distance in T between pairs of elements of $L(q)$. For each $q \in q(T)$, q is included in $Q_{geo}(T)$ with probability $p_q = (diam(q, T))^{-3}$. Note that the membership of $Q_{geo}(T)$ is determined stochastically.

I performed a small simulation study using the source trees from the 100-taxon datasets described in Section 5.3. QMC was applied to four different quartet

encodings of $\mathcal{T}$, the set of source trees:

$$\begin{aligned} Q_{all}(\mathcal{T}) &:= \bigcup_{T \in \mathcal{T}} q(T) \\ Q_{geo+toposhort}(\mathcal{T}) &:= \bigcup_{T \in \mathcal{T}} Q_{geo}(T) \cup Q_{toposhort}(T) \\ Q_{5\text{-}short}(\mathcal{T}) &:= \bigcup_{T \in \mathcal{T}} Q_{5\text{-}short}(T) \\ Q_{25\text{-}short}(\mathcal{T}) &:= \bigcup_{T \in \mathcal{T}} Q_{25\text{-}short}(T) \end{aligned}$$

I found that the accuracy of QMC increased as the number of quartets included in the encoding of the source trees increased, with QMC applied to $Q_{all}(\mathcal{T})$ being the most accurate method (see Figures 5.1 and 5.2) for the 100-taxon datasets. Due to memory constraints, Q_{all} could not be computed for the 500-taxon datasets. I was able to run QMC with the $Q_{geo+toposhort}$ selection method (not shown), and obtained unacceptably high error rates, similar to those achieved by that method on the 100-taxon datasets. In summary, QMC has good performance only when applied to very dense quartet encodings of source trees, however, such encodings are not feasible for large scale supertree problems. Thus, I found that QMC, by itself, can be either topologically accurate or scaleable, but not both.

5.1.2 SCM as a General Supertree Method

SCM produces a supertree from a set of source trees by merging pairs of trees until there is only a single tree left. The merger of two trees begins with the strict consensus [Day, 1985] of the induced trees on the intersection of the leaf sets of the two trees being merged. The remaining taxa in the union of the leaf sets are added to this consensus tree such that they do not contradict any relationships in any of the trees; I define this addition process formally below. The strict consensus merger T of two trees T_1 and T_2, such that $|L(T_1) \cap L(T_2)| \geq 3$, is defined formally as follows

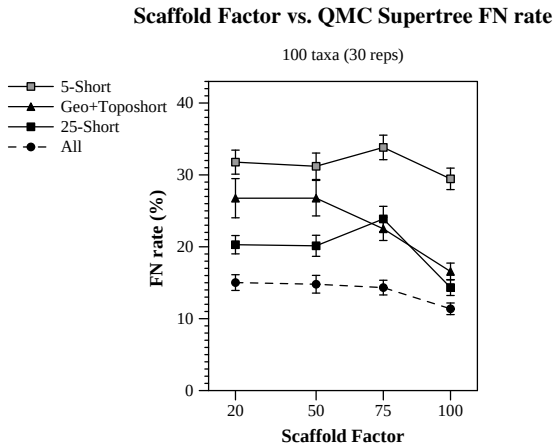

Figure 5.1: False negative (FN) rate (mean with standard error bars) for QMC supertree reconstructions on various quartet encodings as a function of the scaffold factor.

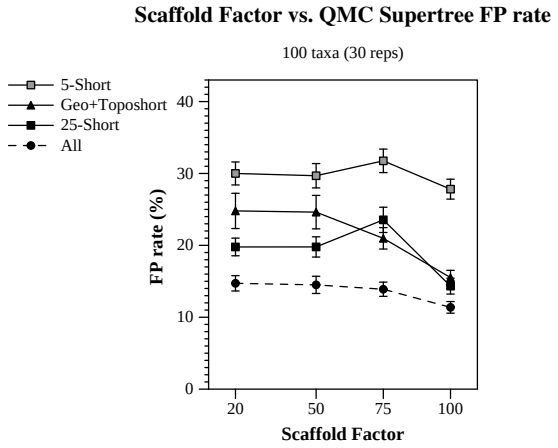

Figure 5.2: False positive (FP) rate (mean with standard error bars) for QMC supertree reconstructions on various quartet encodings as a function of the scaffold factor.

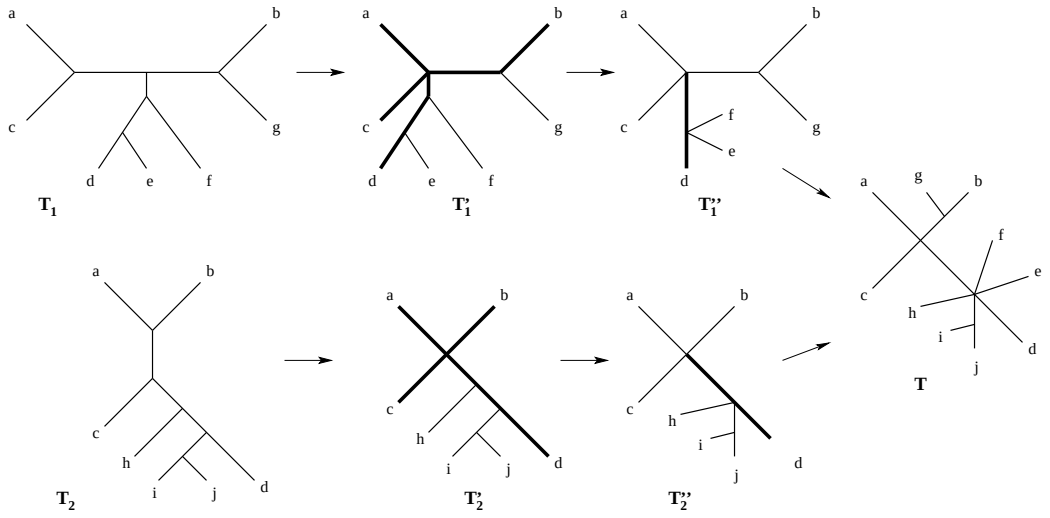

Figure 5.3: SCM merger of two trees T_1 and T_2. In T_1' and T_2', the strict consensus of T_1 and T_2 restricted to their common leaf sets is shown in bold. In T_1'' and T_2'', the edges that are involved in a collapsing of a path in T_1' or T_2' are shown in bold. T is the final SCM tree of T_1 and T_2.

(depicted in Figure 5.3).

Let $X = L(T_1) \cap L(T_2)$, and let T_1' and T_2' be maximally refined trees such that $T_1' \leq T_1$, $T_2' \leq T_2$, and $T_1'|_X = T_2'|_X$. Let $T' = T_1'|_X = T_2'|_X$. (Note that T' is the strict consensus tree of $T_1|_X$ and $T_2|_X$.) Then for edge $e \in E(T')$ that corresponds to a path of length greater than one in both T_1' and T_2', edges of T_1 and T_2 are collapsed in following manner. Let $e_1, ..., e_l$ be the path in T_1' that corresponds to e in T'. Collapse all edges e_i in this path such that, $1 < i < l$. Rename the common vertex of e_1 and e_l, v_e. Use the same process to reduce the corresponding path in T_2' to have length two. After performing this process for each edge of T', merge the resulting collapsed trees T_1'' and T_2'' into a single tree T using the following process. For each edge $e \in T'$ that corresponds to a path of length greater than one in both T_1 and T_2, subdivide e with a vertex v_e and attach to that vertex any subtree of T_1'' and T_2'' that is not in T' and is rooted at v_e. For all other edges of T' subdivide the edge

with as many nodes as necessary and attach the appropriate subtrees from either T_1 or T_2. Notice that by construction, $T|_{L(T_1)}$, and $T|_{L(T_2)}$ are simply contractions of T_1 and T_2, respectively; therefore $T|_{L(T_1)} \leq T_1$, and $T|_{L(T_2)} \leq T_2$.

In the rest of this chapter, I refer to "SCM supertrees" of sets of more than two trees, which simply refers to the result of consecutive strict consensus mergers of pairs of trees, such that each pair of trees being merged have at least three taxa in common. I pause here to present some theoretical results that will be useful in the SuperFine algorithm.

Theorem 5.1.1. *Let $\mathcal{T}$ be a collection of trees and let T be a SCM supertree of $\mathcal{T}$. Then for every $t \in \mathcal{T}$, $\Sigma(T|_{L(t)}) \subseteq \Sigma(t)$.*

Proof. This result is proved by induction on cardinality of $\mathcal{T}$. By construction, the result holds for $|\mathcal{T}| = 2$. Assume $|\mathcal{T}| = k$, and that the result holds for sets of $k-1$ trees. Let t be an member of $\mathcal{T}$, and consider the last two trees T_1 and T_2 to be merged to create the final tree T. Then one of the following must be true: $t = T_1$, $t = T_2$, or either T_1 or T_2 is the strict consensus merger of some set of trees $\mathcal{T}'$ that includes t. Our base case shows that $\Sigma(T|_{L(T_1)}) \subseteq \Sigma(T_1)$, and $\Sigma(T|_{L(T_2)}) \subseteq \Sigma(T_2)$. Thus, if $t = T_1$ or $t = T_2$, then $\Sigma(T|_{L(t)}) \subseteq \Sigma(t)$. Now suppose, without loss of generality, that T_1 is the strict consensus merger of a set of trees $\mathcal{T}'$ such that $t \in \mathcal{T}'$. Then $|\mathcal{T}'| < k$ and by the induction hypothesis $\Sigma(T_1|_{L(t)}) \subseteq \Sigma(t)$, and the result follows. □

The following corollary of Theorem 5.1.1 is immediate.

Corollary 5.1.2. *Let $\mathcal{T}$ be a collection of trees, let T be a SCM supertree of $\mathcal{T}$, and let v be a vertex of T. Let T' be a subtree of T rooted at a vertex u adjacent to v, such that v is not a vertex of T'. Then for any $t \in \mathcal{T}$, one of the following three conditions holds: $L(t) \subseteq L(T')$, $L(t) \cap L(T') = \emptyset$, or $L(t) \cap L(T') | L(t) - L(T') \in \Sigma(t)$*

As I will show in Section 5.2, this corollary is used in the second step of the SuperFine algorithm.

The topology of the supertree returned by SCM depends on the order in which the trees are merged. SCM is most commonly used in Disk Covering Methods (DCMs), a particular class of divide and conquer approaches designed to boost the performance of standard phylogenetic reconstruction methods [Huson et al., 1999a,b, Nakhleh et al., 2001, Warnow et al., 2001, Tang and Moret, 2003, Roshan et al., 2004b]. A DCM uses some criterion to break the full dataset into overlapping subproblems, applies a phylogenetic reconstruction method to each subproblem, and then uses SCM to combine the resulting trees into a single phylogeny on the full dataset. In the context of DCMs, the best order in which to merge the subproblems is known and depends on the method used to divide into subgroups.

I was interested in using SCM as a general supertree method where the characteristics of the subproblems are governed by biological processes and the systematic practices of biologists. In this case, the best order in which to merge the subproblems is unknown, and therefore, one must decide the order in which to merge the source trees. I explored several different ways of choosing an order for the pairwise merger of subtrees. Below, I describe several alternative approaches to deciding the merger order, and present the results of a small simulation study testing the performance of SCM under these ordering methods.

One way that the SCM of two trees could be poorly resolved is if the topology of the induced trees on the overlapping leaf set are quite distant from each other in terms of the bipartitions they contain. Another is if the two trees have a relatively small number of leaves in common. Since, in the context of general supertree methods, one cannot change the leaf sets or topology of the trees, I focus on minimizing the loss of resolution by choosing a good order in which to merge the trees. I devised three methods methods that try to maximize the taxonomic overlap and one that

attempts to minimize the number of taxa not in the overlap.

- **Maximum Backbone.** The **backbone number** between two trees is the number of common leaves (or taxa). In the Maximum Backbone method, the two trees with the most taxa in common, i.e., the greatest backbone number, are removed from the set of source trees, merged, and the resulting tree is added back into the set of source trees. If there is a tie, then the ordering picks the first pair found. This process is repeated until there is only one tree.

- **Backbone Link.** The Backbone Link method is a variant of Maximum Backbone ordering. This method begins by computing the backbone number for each pair of trees in the set of source trees. As with Maximum Backbone, in this method begins by first selecting the pair of trees with the greatest backbone number (selecting the first pair found if there is a tie). The two trees are deleted from the set of source trees and added to an ordered list of trees. From this point forward, the most recent resulting SCM tree is always used in the next merger, and is merged with a tree from the remaining set of source trees that has the greatest overlap. This process is repeated until the set of source trees is empty.

- **Normalized Maximum Backbone.** The Normalized Maximum Backbone method follows the same procedure as the Maximum Backbone method, but uses the number of taxa in the overlap of two trees divided by the union of their leaf sets as the value to maximize between consecutive pairs of trees.

- **Minimum Non-Backbone.** The Minimum Non-Backbone method seeks to minimize the number of taxa that are in either of the two trees but not in both. I refer to this as the non-backbone number. This method uses the same algorithm as the Maximum Backbone, but rather than computing the backbone number, this order is based on the non-backbone number.

The running time of the order selection methods depends on the maximum number of taxa in a single source tree, which is bounded by the number n of taxa in the full dataset and the number k of source trees. The running time analyses for all four methods is trivial, and the proofs are omitted.

Theorem 5.1.3. *The Maximum Backbone, Normalized Maximum Backbone, and Minimum Non-Backbone all run in $O(k^2 n \log n + k^3 n)$ time, while Backbone Link runs in $O(kn \log n + k^3 n)$ time.*

Under each of these four orderings, SCM runs in a matter of minutes for even the 1000-taxon datasets (for details see Section 5.4).

I compared the performance of SCM with these orderings in a small simulation study, this time on all datasets described later in Section 5.3. I found that SCM is a conservative method that often produces highly unresolved supertrees with low false positive rates and high false negative rates (see Figures 5.4 and 5.5). The false positive rate of SCM is virtually the same for all the ordering methods (Fig. 5.4), with Maximum Non-Backbone having a slightly lower FP rate than that of the other orderings. With respect to false negative rate, Maximum Backbone, Normalized Maximum Backbone, and Backbone Link, have the best performance (Fig. 5.5).

5.2 The SuperFine Algorithm

The SuperFine algorithm proceeds in two phases: 1) compute a supertree from source trees, using SCM, and 2) repeatedly resolve each polytomy in the SCM supertree, where a polytomy is a node of degree greater than three. Resolving a single polytomy involves first encoding each source tree as a set of quartets, applying QMC to that set of quartets, and finally resolving the polytomy based on the tree returned by QMC.

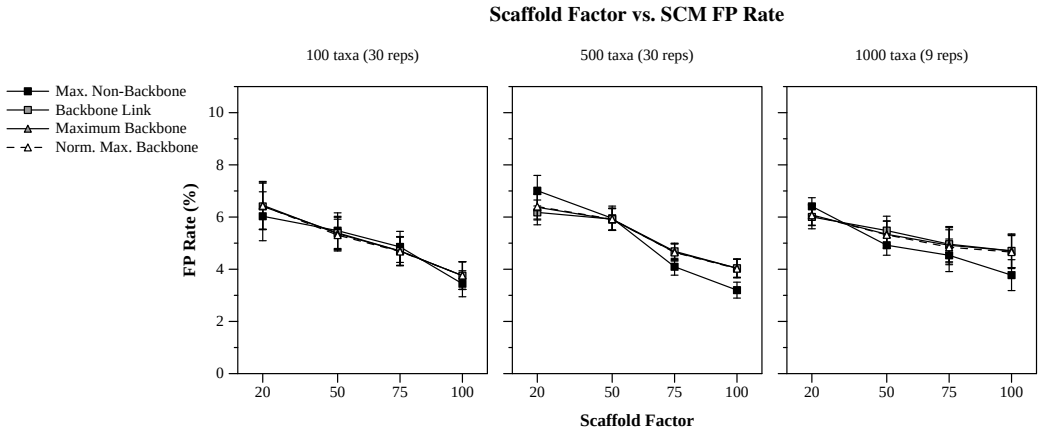

Figure 5.4: False positive (FP) rate (mean with standard error bars) for SCM supertree reconstructions, applied using various ordering methods, as a function of the scaffold factor.

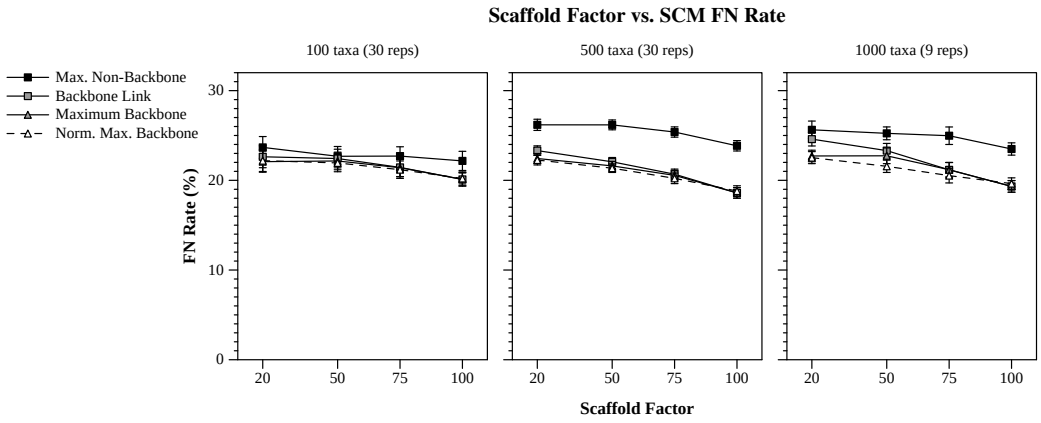

Figure 5.5: False negative (FN) rates (mean with standard error bars) for SCM supertree reconstructions, applied using various ordering methods, as a function of the scaffold factor.

5.2.1 Step 1: Computing the SCM Tree

To compute the SCM supertree, the Maximum Backbone order selection method is used. I chose this order selection method because its false negative and false positive rates were among the best for the various ordering methods.

5.2.2 Step 2: Refining Polytomies

The next step of this method iterates over the polytomies in the SCM supertree, resolving each polytomy based on the topology of the source trees. Let $\mathcal{T}$ be a set of source trees and let T be a supertree computed using some ordering of pairwise strict consensus mergers, such that each merger pair has at least three leaves in common. Let v be a node of degree d in T, such that $d \geq 4$, i.e., v is a polytomy node of T. The given polytomy, v is resolved, producing a tree T' that is a refinement of T, using the following procedure.

1) Root T at v, and let $v_1, ..., v_d$ be the children of v and $T_1, ..., T_d$ be the subtrees rooted at $v_1, ..., v_d$, respectively

2) Compute a set $\mathcal{T}_v$ of relabeled and reduced source trees based on the subtrees of T rooted at nodes adjacent to v. Let $\phi : L(T) \rightarrow \{1, ..., d\}$ be defined by $\phi(x) = i$ for $x \in L(T_i)$. Notice that for every $x \in L(T)$, x is a leaf in exactly one of these d subtrees, thus ϕ is well-defined. Using this mapping, relabel the leaves of the source trees. For each source tree, recursively replace any sibling pairs (or perhaps groups of sibling taxa if a source tree is non-binary) that share the same label with a single leaf with that label. Corollary 5.1.2 implies that applying this process to one of the source trees will result in a tree with at most one leaf with each label (see Lemma 5.2.1 below). Thus, each member of $\mathcal{T}_v$ is a phylogenetic tree whose leafset is a subset of $\{1, ..., d\}$.

3) Compute $Q_{all}(\mathcal{T}_v)$ as defined in Section 5.1.1.

4) Apply QMC to $Q_{all}(\mathcal{T}_v)$ to obtain a tree T^* leaf-labeled by the set $\{1, 2, ..., d\}$.

5) Construct T' by grafting each T_i onto T^*, attaching the root of T_i to leaf i in T^*, for each $i \in \{1, ..., d\}$.

Note that, if an edge e of T^* induces the bipartition $A|B$ of $\{1, ..., d\}$, then $X|Y$, such that $X = \{x \in L(T) : \phi(x) \in A\}$ and $Y = \{y \in L(T) : \phi(x) \in B\}$, is the bipartition induced by e in T'. Therefore, the last step in the above process is equivalent to inserting additional edges into T. It follows that the order in which the polytomies are resolved does not affect the final topology of the supertree.

The following theorem is an immediate result of Corollary 5.1.2.

Lemma 5.2.1. *Let $\mathcal{T}$, T, v, and ϕ be as in the description above. Then for any $i \in \{1, ..., d\}$ and $t \in \mathcal{T}$, one of the following conditions holds: $L(t) \subseteq \phi^{-1}(i)$, $L(t) \cap \phi^{-1}(i) = \emptyset$, or $L(t) \cap \phi^{-1}(i)|L(t) - \phi^{-1}(i) \in \Sigma(t)$.*

With this result, one can easily see that source trees relabeled and collapsed using the process described in 2) above can have at most one leaf with each label.

5.3 Performance Evaluation Methods

I evaluate the performance of our new method based on topological accuracy, running time, and similarity to the source trees. SuperFine is applied to source tree datasets generated using SMIDGen described in Chapter 3 and the performance of SuperFine is compared to that of SCM, MRP, and combined analysis.

5.3.1 Data Generation

For source tree datasets, a subset of those used in the study in Chapter 4 are used here. In that chapter, I found that the factors that had the greatest effect on supertree accuracy were, listed in order from greatest effect to least, base method

(MP vs. ML), the density of the scaffold tree (scaffold factor), and the number of taxa in the supertree. I am also interested in comparing our new supertree method to the best overall supertree method which, in Chapter 4, was shown to be MRP on ML source trees. Included in the tests in this chapter is the full range of values, for number of taxa and for scaffold factors, explored in Chapter 4. Thus, the datasets used here are as follows (see Chapters 3 and 4 for details).

- Model trees (number of replicates) with number of clade-based studies for the given dataset size:
  - 100 taxa (30 reps) with five clade-based studies
  - 500 taxa (30 reps) with 15 clade-based studies
  - 1000 taxa (10 reps) with 25 clade-based studies
- Clade-based studies: each clade based study is based on three non-universal genes. All gene sequences are of length 500.
- Scaffold factors:
  - 20%
  - 50%
  - 75%
  - 100%
- Number of scaffold genes: four (each universal and of length 500)
- Source tree reconstruction method: RAxML in default GTRMIX setting

Since, in addition to supertree methods, the performance of SuperFine is compared to that of combined analysis methods, combined data sets are also constructed; as in Chapter 4, I use the concatenated source tree datasets. See figures in Section 4.2 for average percent missing data in the combined dataset for a given model condition.

To each of the source tree datasets described above, the following supertree methods are applied:

- **SuperFine.** SuperFine applied as described in Section 5.2.
- **SCM.** Strict consensus merger applied using the merger order described in Section 5.2.
- **MRP.** The MRP supertree is the majority consensus of the best trees found using a PAUP* ratchet with 100 ratchet iterations (see Section 3.2.4 for details).

The analyses of the combined datasets use RAxML in its GTRMIX default setting for the 100- and 500-taxon data sets and in its GTRCAT default setting for the 1000-taxon data sets.

5.3.2 Supertree Accuracy and Quality Measures

I measure topological accuracy using the false positive and false negative rates of the estimated trees against the model trees. In this study, I also investigate the distance of the supertrees to the source trees. I compare each source tree t to the induced subtree $\hat{T}|_{L(t)}$ of the supertree $\hat{T}$ on the leaf set $L(t)$. I compute the distance from source trees to the induced subtrees of the supertrees by summing the false positives and summing the false negatives to each source tree. Let $\mathcal{T}$ be a set of source trees and $\hat{T}$ be a supertree with leafset $L(\mathcal{T})$. I then define the following distance measures:

$$Sum\text{-}of\text{-}FN = \sum_{t \in \mathcal{T}} FN(\hat{T}|_{L(t)}, t)$$
$$Sum\text{-}of\text{-}FP = \sum_{t \in \mathcal{T}} FP(\hat{T}|_{L(t)}, t)$$

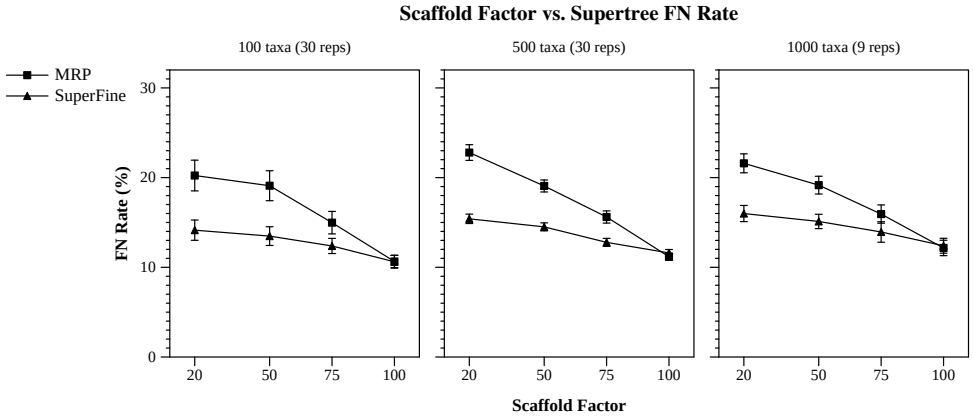

Figure 5.6: False negative (FN) rates (mean with standard error bars) for SuperFine and MRP supertree reconstructions as a function of the scaffold factor.

I also compare the estimated trees based on running time and percent resolution, where the percent resolution of a tree T is $\frac{|E^{\circ}(T)|}{|L(T)|-3} \cdot 100$.

5.4 Performance of SuperFine

In this section, SuperFine is compared to MRP, SCM, and combined analysis based on topological accuracy, the percent resolution, the distance to the source trees, and combined analysis trees, and the running time of MRP to SuperFine. The results of this simulation study indicate that SuperFine produces a more topologically accurate than MRP, particularly with respect to false negative rates. SuperFine approaches the accuracy of combined analysis under most model conditions. I also show that SuperFine produces these trees in significantly less time than that of MRP.

Topological accuracy of SuperFine

When comparing SuperFine to MRP based on FN rates, SuperFine outperforms MRP under all model conditions (Fig. 5.6). The advantage of SuperFine over MRP increases as the scaffold factor decreases, with MRP having an FN error rate that

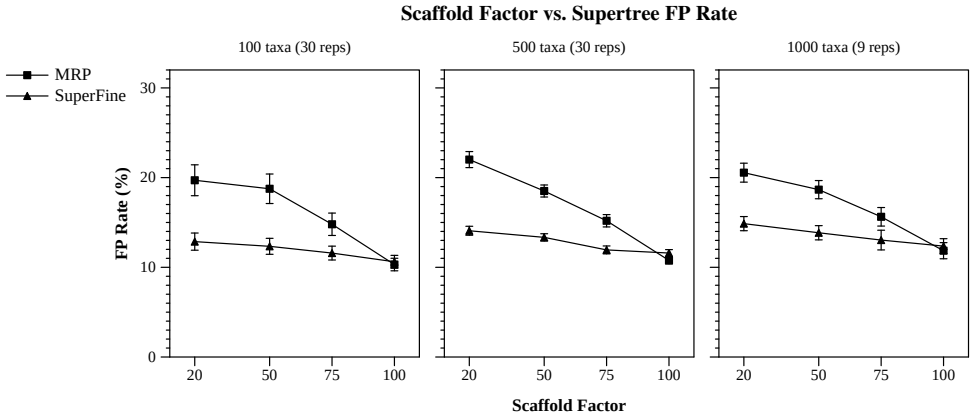

Figure 5.7: False positive (FP) rates (mean with standard error bars) for SuperFine and MRP supertree reconstructions as a function of the scaffold factor.

is around 50% higher than that of SuperFine on datasets with 20% scaffold factors.

With respect to FP rates, SuperFine is competitive with MRP under most model conditions (Fig. 5.7). MRP, however, has a lower false positive rate than SuperFine for model conditions with 100% scaffold factors and for the datasets with 1000 taxa and a scaffold factor of 75%. Note that to obtain optimal trees according to false positives rates, one need only return a "star" tree, a tree with zero internal edges. Therefore, while it is important to obtain trees with both low FP and low FN rates, obtaining low FN rates is more significant.

As mentioned in Chapter 4, the maximum likelihood combined analysis trees are fully resolved, and as I will show later in this section, the SuperFine trees are nearly fully resolved. One do not, therefore, see much of a change in the relative performance of SuperFine and combined analysis whether one measure topological accuracy with FP or FN. SuperFine remains competitive with combined analysis, with respect to FN rate, under model conditions with 20% scaffold factors. This provides a distinct contrast to the comparison of MRP to combined analyses, where MRP only approaches the accuracy of a combined analysis when the scaffold factor

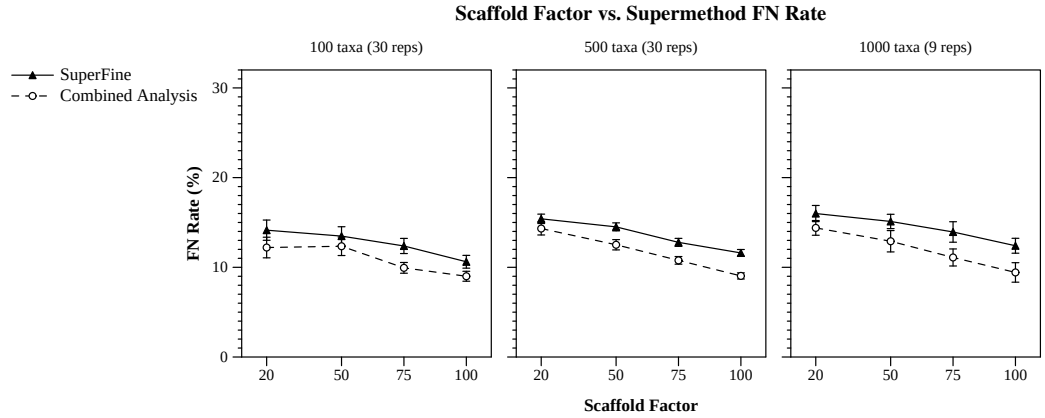

Figure 5.8: False negative (FN) rates (mean with standard error bars) for SuperFine supertree and combined analysis reconstructions as a function of the scaffold factor.

is 100%. With respect to FP rates, Superfine is on par with combined analysis for the datasets with sparser scaffolds, and less accurate than combined analysis for datasets with 75% and 100% scaffolds (Fig. 5.9).

SuperFine achieves significantly lower FN rates than SCM alone (Fig. 5.10). This indicates that in the refinement step of SuperFine, a significant portion of true branches are recovered. However, given the rise in the FP rate from the SCM trees to the SuperFine trees (Fig. 5.11), the refinement step is clearly adding false edges as well. As stated earlier, while one would like to minimize the number of false positives in the supertree, this is not a good optimization criterion, and thus I focus on reducing the false negative rate.

Percent resolution of SuperFine

It is clear from Figure 5.12, that the refinement step of the SuperFine method is producing nearly fully resolved trees from the trees it computes in the SCM step. In fact, given source tree datasets containing a complete tree, the SuperFine tree is fully resolved. Note that, in this study, the source trees are binary. If the

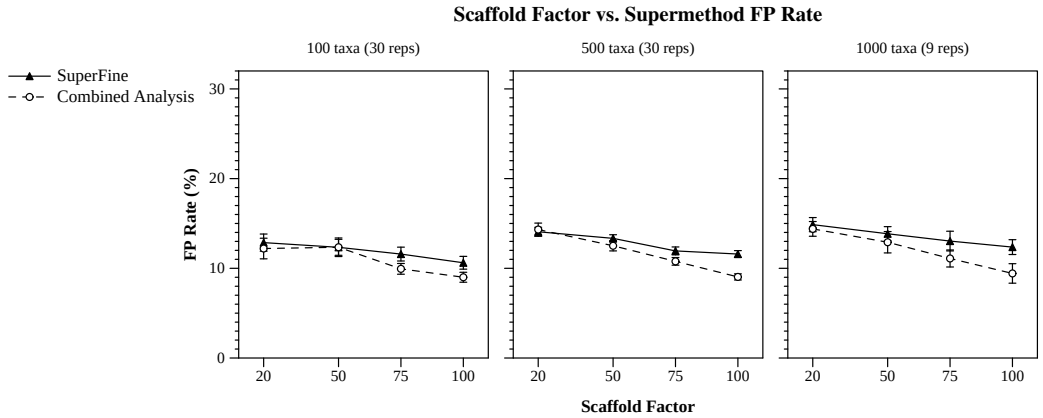

Figure 5.9: False positive (FP) rates (mean with standard error bars) for SuperFine supertree and combined analysis reconstructions as a function of the scaffold factor.

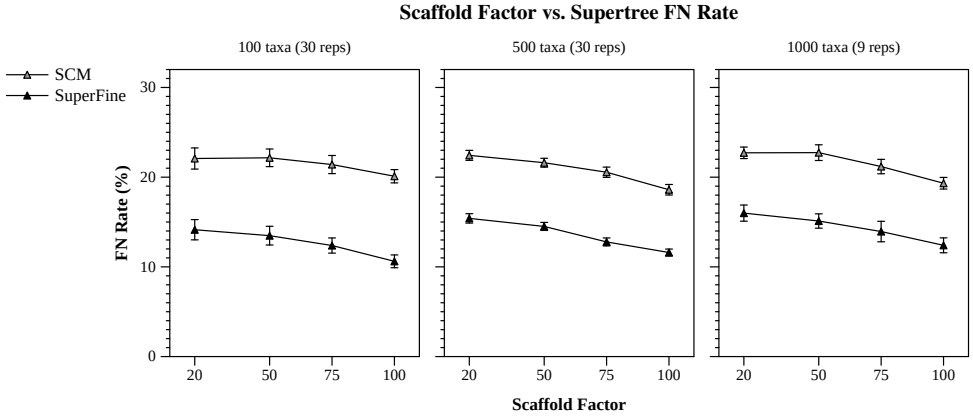

Figure 5.10: False negative (FN) rates (mean with standard error bars) for SuperFine and SCM supertree reconstructions as a function of the scaffold factor.

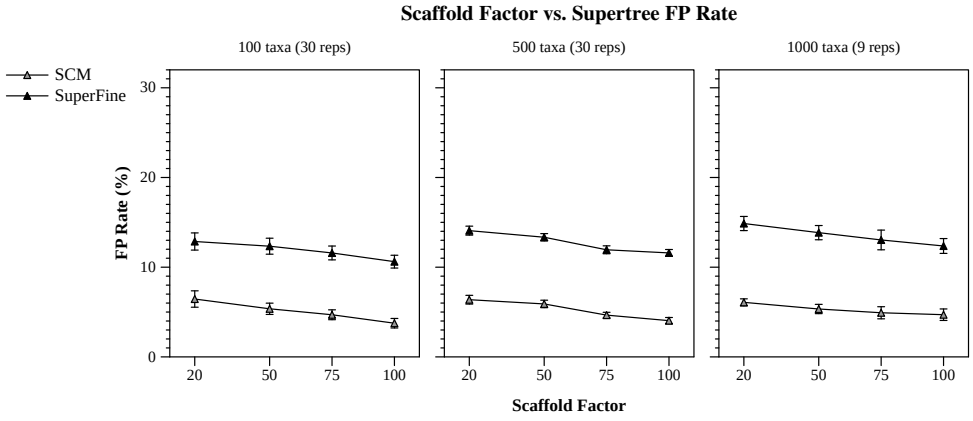

Figure 5.11: False positive (FP) rates (mean with standard error bars) for SuperFine and SCM supertree reconstructions as a function of the scaffold factor.

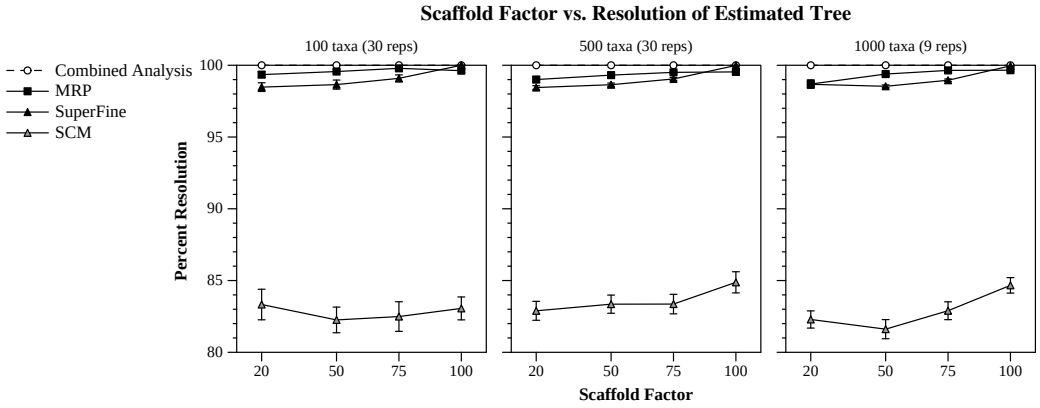

Figure 5.12: Percent resolution (mean with standard error bars) of supertree and supermatrix reconstructions as a function of the scaffold factor.

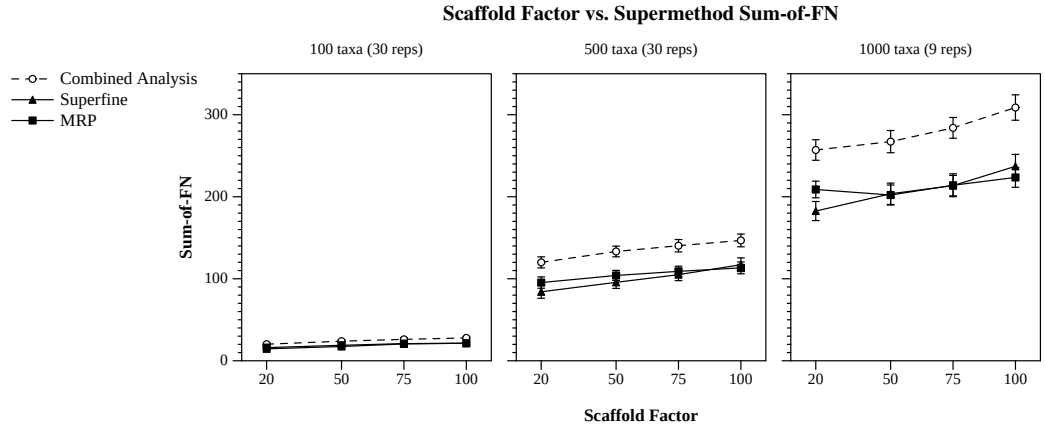

Figure 5.13: Sum of the false negatives (Sum-of-FN) (mean with standard error bars) from supertree reconstructions to the source trees as a function of the scaffold factor.

compatibility of the source trees was high, one would expect the compatibility of the source trees, one would expect the resolution of SuperFine to decrease if the source trees were unresolved. As mentioned earlier, the combined analysis trees are fully resolved. MRP produces supertrees that are slightly more resolved trees than SuperFine, except for datasets with 100% scaffold factor. For datasets with 100% scaffold factor, SuperFine produces fully resolved supertrees, while MRP is nearly fully resolved.

Distance to the source trees

SuperFine and MRP have similar Sum-of-FN scores, both lower than that of combined analysis (see Fig. 5.13). Given that combined analysis is the most topologically accurate method, its higher Sum-of-FN distance to the source trees indicates that minimizing that distance is a poor optimization criterion for the given input data. Recall, however, that the source trees used in this simulation study have high error rates (see Figure 4.7 in Chapter 4). Thus, it is possible that these high error rates are what causes minimizing the Sum-of-FN to be a poor optimization criterion.

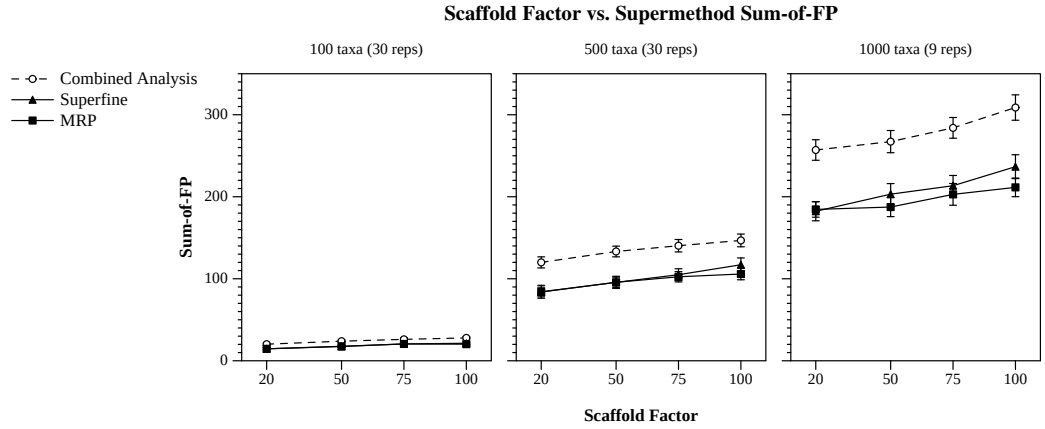

Figure 5.14: Sum of the false positives (Sum-of-FP) (mean with standard error bars) from supertree reconstructions to the source trees as a function of the scaffold factor.

If one had source trees that were more topologically accurate, this might in fact be a good optimization criterion.

When using the Sum-of-FP distances, the relative distances to the collection of source trees from the SuperFine, MRP, and combined analysis trees are similar to that measured by Sum-of-FN (Fig. 5.13). This indicates that this too is a poor optimality criterion under our model conditions. For the 1000-taxon datasets, MRP returns supertrees that are slightly more similar to the source trees in terms of Sum-of-FP than the other two methods.

Running time

Superfine runs in significantly less time than MRP (Table 5.2), and the difference in running time between the two methods increases as the number of taxa in the model tree increases. Thus if one already has source trees, and only needs to compute a supertree, SuperFine is much faster than MRP. If however, one includes the time to compute the source trees, there is not much difference between SuperFine and MRP, but both supertree methods take much less time than computing a combined

Number of Taxa	Scaffold Factor	SuperFine	MRP	Combined Analysis
100	20	0:10:05 (0:00:02)	0:10:14 (0:00:12)	0:21:08
	50	0:07:40 (0:00:02)	0:07:46 (0:00:09)	0:15:24
	75	0:08:13 (0:00:02)	0:08:19 (0:00:08)	0:15:26
	100	0:09:26 (0:00:03)	0:09:32 (0:00:09)	0:15:18
500	20	0:58:54 (0:00:22)	1:04:27 (0:05:56)	10:01:17
	50	1:09:41 (0:00:27)	1:15:26 (0:06:12)	9:47:01
	75	1:20:03 (0:00:30)	1:26:13 (0:06:40)	9:50:36
	100	1:38:55 (0:00:34)	1:44:52 (0:06:32)	8:41:31
1000	20	2:26:21 (0:01:28)	3:12:12 (0:47:19)	49:50:05
	50	3:11:47 (0:01:54)	3:50:40 (0:40:47)	45:42:31
	75	3:49:14 (0:02:09)	4:28:51 (0:41:47)	47:27:17
	100	5:09:13 (0:02:22)	5:47:34 (0:40:43)	46:19:36

Table 5.1: Average running time for each of the three super methods. Running times are given in hours:minutes:seconds. For the two supertree methods, the time to compute just the supertree is given in parentheses following the full running time, which includes the time taken to compute source trees.

analysis (Table 5.2).

The two steps of SuperFine, computing the SCM supertree and resolving polytomies, can make up varying proportions of the running time of SuperFine, depending on the size of the model tree. For the 100-taxon datasets, resolving the polytomies takes approximately the same amount of time as computing the SCM supertree. For the 500-taxon datasets, the refining step takes two to three times longer than computing the SCM supertree. For the 1000-taxon datasets, the refining step takes three to five times longer on average. Thus, it seems that as the number of taxa increases, the running time of the refinement step makes up a larger portion of the running time of SuperFine.

Number of Taxa	Scaffold Factor	SCM	Refining Polytomies	Total
100	20	00:01.2	00:00.9	00:02.1
	50	00:01.1	00:01.1	00:02.2
	75	00:01.2	00:01.2	00:02.4
	100	00:01.3	00:01.3	00:02.6
500	20	00:05.9	00:16.5	00:22.4
	50	00:06.7	00:19.8	00:26.6
	75	00:07.9	00:22.0	00:29.8
	100	00:09.9	00:24.4	00:34.3
1000	20	00:13.7	01:14.3	01:28.0
	50	00:23.1	01:30.8	01:54.0
	75	00:29.3	01:39.9	02:09.2
	100	00:37.1	01:44.9	02:22.0

Table 5.2: Average running time for the two phases of SuperFine: 1) computing the SCM tree and 2) refining the SCM tree using QMC. Running times are given in minutes:seconds.

5.5 Conclusions and Future Work

In this chapter, I have shown that SuperFine generates more accurate supertrees than MRP under most conditions tested here. In particular, Superfine outperforms MRP on datasets with sparse scaffolds, a condition under which supertree methods are, potentially, most appealing. Furthermore, SuperFine narrows the gap in accuracy between supertree methods and combined analysis. The relatively short running time of this new method, in comparison to MRP, is also an attractive feature.

I plan to explore other versions of this method using alternative quartet encodings of the source tree and different ways of applying QMC to those sets of quartets. As there are multiple processes in our method that are stochastic, I am also interested in exploring the performance of a version that runs many replicates of SuperFine and returns a consensus tree as well as confidence values for the branches of the supertree. I will also compare the accuracy of SuperFine to that of an MRP

tree computed using the greedy consensus instead of the majority consensus. A greedy consensus MRP supertree would likely have a lower false positive rate and thus be more competitive with SuperFine than the majority consensus MRP trees.

Chapter 6

Closure Sets of Rooted Triples

6.1 Introduction

A desirable property of supertree methods, mentioned in Chapter 2, is that if the set of source trees T is compatible, the supertree returned must display T, i.e. the supertree is an element of $co(T)$. Requiring that a supertree display a particular set of trees implies it must display other relationships, those contained in every element of $co(T)$, which may not be contained in any source tree. *Closure operations* gather up the collection of relationships contained in or implied by a set of trees. Such operations can even be defined for incompatible sets of trees, and can, in fact, be used to discover conflicts in sets of trees.

This chapter, explores closure operations on sets of rooted triples. In Section 6.2 I formally define the closure of a set of rooted triples and propose a family of supertree methods based on an extension of the desirable property mentioned above. In Section 6.3, I determine, for any phylogenetic tree, the minimum number of rooted triples whose closure gives all the rooted triples for that tree. In Section 6.4, I discuss a possible application of the result in Section 6.3. The results in sections 6.2 and 6.3 were developed jointly with Stefan Grünewald and Mike Steel,

and were published in Grünewald et al. [2007].

6.2 Background

Given a collection R of rooted triples, let $L(R)$ denote the set of leaf labels that appear in at least one element of R and, in keeping with the notation introduced in Chapter 2, let $co(R)$ denote the set of rooted phylogenetic trees on $L(R)$ that display all the trees in R. As in the context of general collections of trees, we say the collection of rooted triples R is *compatible* if $co(R)$ is non-empty. Recall that $r(T) = \{ab|c : a, b, c \in T \text{ and } T|_{\{a,b,c\}} = ab|c\}$.

6.2.1 Closure of a Set of Rooted Triples

We write $R \vdash ab|c$ if R is compatible and both $R \cap \{ac|b\}$ and $R \cap \{bc|a\}$ are incompatible, which is equivalent to requiring that every tree that displays R also displays $ab|c$.

If R is a compatible set of rooted triples, we define the *closure of* R, $cl(R) = \bigcap_{T \in co(R)} r(T)$. Equivalently one can define $cl(R)$ as the set $\{ab|c : R \vdash ab|c\}$.

This operation satisfies the usual three properties of a closure operator, namely: $R \subseteq cl(R)$; $cl(cl(R)) = cl(R)$ and if $R_1 \subseteq R_2$ are compatible, then $cl(R_1) \subseteq cl(R_2)$.

If R is not compatible, then one can also define a closure of R as follows. We say that a set (compatible or not) R^* is *closed* if $\forall R' \subseteq R^*$ such that R' is compatible, $cl(R') \subseteq R^*$. In particular, the set $R(X)$ of all $3\binom{n}{3}$ rooted triples on X is closed, and so given a set $R \subseteq R(X)$ we can define the *closure* of R, denoted $Cl(R)$ to be the intersection of all closed sets containing R. This also satisfies the three properties of a closure operator, and when R is compatible we have $Cl(R) = cl(R)$.

6.2.2 An Axiomatic Requirement of Supertree Methods

In Golobof and Pol [2002], a desirable requirement of supertree methods was described. The authors described this requirement, semi-formally, as follows: "the property of [the output tree] displaying $x|yz$ if it is found in some input tree or implied by some combination of input trees and no input tree or combination of input trees displays or implies $y|xz$ or $z|xy$".

Here we present a possible formalization of this requirement and determine whether it can be achieved. First we need the following lemma, which is a slight strengthening of Proposition 9(2) of Bryant and Steel [1995], and is established by the same argument used in that result.

Lemma 6.2.1. *Let R be a set of rooted triples. R is incompatible if and only if $\exists\ R' \subset R$ such that $\forall\ ab|c \in R - R'$ either $R' \vdash ac|b$ or $R' \vdash bc|a$.*

Proof. If R is compatible, then $cl(R)$ is compatible. Then for any $R' \subset R$ and rooted triple $ab|c$ such that $R' \vdash ab|c$, $ab|c \in cl(R') \subset cl(R)$. Thus $ab|c$ cannot contradict any rooted triple in R. Now suppose R is incompatible. Let R' be a maximal compatible subset of R and let $ab|c$ be an element of $R - R'$. Then $Cl(R) \cup ab|c$ is incompatible, thus by Proposition 9(1) of Bryant and Steel [1995], either $ac|b \in Cl(R')$ in which case $R' \vdash ac|b$, or $bc|a \in Cl(R')$ in which case $R' \vdash bc|a$. □

Given two sets of rooted triples R_1, R_2, let

$$[R_1, R_2] := \{ab|c \in R_1 : \text{there does not exist } R' \in R_2 \ s.t.\ R' \vdash ac|b \text{ or } R' \vdash bc|a\}.$$

Proposition 6.2.2. *Let R_1 and R_2 be any set of rooted triples such that $R_1 \subseteq R_2$, then $[R_1, R_2]$ is compatible. In particular $[R_1, R_1]$ is compatible.*

Proof. Suppose $[R_1, R_2]$ is incompatible. By Lemma 6.2.1 there exists a set $R' \subseteq [R_1, R_2] \subseteq R_1 \subseteq R_2$ and $ab|c \in [R_1, R_2] - R' \subseteq R_1$ such that either $R' \vdash ac|b$ or

$R' \vdash bc|a$. However, this implies that $ab|c \notin [R_1, R_2]$, a contradiction. □

Proposition 6.2.2 suggests a family of supertree methods. Given as input a collection $\mathcal{T}$ of rooted phylogenetic trees, first take the set $r(\mathcal{T})$ and from it, construct two sets $R_1 \subseteq R_2$. Subsequently, construct the Aho tree, described below, in Section 6.2.3, of $[R_1, R_2]$, denoted $\mathcal{A}_{[R_1,R_2]}$. One can obtain a supertree method satisfying the property described by Golobof and Pol [2002] above by setting $R_1 = R_2 = Cl(r(\mathcal{T}))$. One might also consider setting $R_1 = r(\mathcal{T})$ and R_2 to either $r(\mathcal{T})$ or $Cl(r(\mathcal{T}))$, to obtain a supertree with similar desirable properties.

6.2.3 The Aho Algorithm

Given a set of compatible rooted triples R the Aho algorithm uses the graph described below to recursively construct a rooted tree that displays R. (This description of the Aho algorithm is based on that of Semple and Steel [2003]). First, for any set R of rooted triples, we define $G(R)$ to be the graph with vertex set $L(R)$ and with an edge between two vertices a and b exactly when there is some $c \in L(R)$ such that $ab|c \in R$. We now define the Aho algorithm.

Algorithm: AHO(R, v, T)

Input: R, a collection of rooted trees and a vertex v

Output: If R is compatible, output a tree leaf-labeled by the leafset of R rooted at v, and if R is incompatible return "R is incompatible"

1. Let S be the label set of R.

2. If $|S| = 1$, return a single vertex labeled by the one element of S. If $|S| = 2$, return the tree obtained by attaching two vertices labeled by the two elements of R to v. Otherwise, construct $G(R)$.

3. If $G(R)$ has a single connected component, return "R is incompatible". Otherwise, let $\{S_1, ..., S_n\}$ be the vertex sets of the connected components of $G(R)$, and for each i call $Build(R_i, v_i, T_i)$ where R_i is the set of elements of R whose leafsets are contained in S_i. Then return the tree obtained by attaching each tree T_i to v via an edge (v, v_i).

For a compatible set of rooted triples R, let $\mathcal{A}_R$ be the Aho tree on R. We know from one result in Semple [2003] that if R identifies T, $\mathcal{A}_R = T$. We use this result as well as variants of the graph $G(R)$ in the proofs of our main results. .

6.3 Minimal Sets Whose Closure Gives All the Information in a Tree

For every rooted phylogenetic tree T, $r(T)$ is closed and, in general, there exist subsets R of $r(T)$ such that $cl(R) = r(T)$. An interesting question, then, is what is the size of the smallest set of rooted triples R which has the property that $cl(R) = r(T)$. In this section, we compute a tight lower bound on the cardinality of, and show how to construct, such a set R.

Before exploring this further, we first note that this question has an equivalent reformulation. A collection of rooted triples R *identifies* a rooted phylogenetic T if T displays R and every other tree that displays R is a refinement of T. That is, $co(R) = \{T' : T \leq T', L(T) = L(T')\}$. Before we can state the reformulation of our question, we first need the following result:

Lemma 6.3.1. *For any subset R, of $r(T)$ we have $cl(R) = r(T)$ if and only if R identifies T.*

Proof. By definition, $cl(R) = \bigcap_{T' \in co(R)} r(T')$. Thus, $cl(R) = r(T)$ if and only if $r(T) \subseteq r(T')$ for all $T' \in co(R)$, but $r(T) \subseteq r(T')$ if and only if T' refines T.

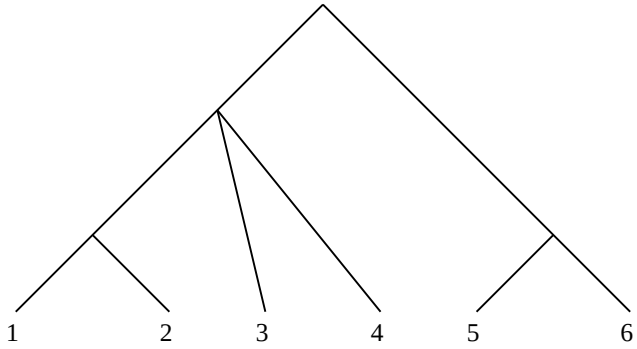

Figure 6.1: A rooted phylogenetic tree T that displays 12|3 and 13|6 but not 13|4 or 15|4.

Therefore, $cl(R) = r(T)$ if and only if T' refines T for every $T' \in co(R)$ which is precisely the requirement for R to identify T. □

In view of Lemma 6.3.1, our problem is to determine the smallest number of rooted triples needed to identify T.

For a rooted tree T and $v \in V(T)$, the set of *descendants of* v, denoted $\mathrm{des}_T(v)$, is the set of leaves that can be reached via a directed path in T starting at v. We write simply $\mathrm{des}(v)$ when T is clear from context.

A rooted triple $ab|c$ *distinguishes* an edge (u, v) in T if and only if $c \in \mathrm{des}(u) - \mathrm{des}(v)$ and there exists $(v, y), (v, z) \in E(T)$ with $y \neq z$ such that $a \in \mathrm{des}(y)$ and $b \in \mathrm{des}(z)$. If R identifies T, then it is clearly necessary that for each internal edge of T, R contains at least one rooted triple that distinguishes that edge. For a binary tree, this condition is also sufficient, and thus for a tree on n leaves, a set of cardinality $n - 2$ (one rooted triple for each of the $n - 2$ internal edges) is all that is needed to identify the tree [Steel, 1992]. As noted in Wilkinson et al. [2004], "calculating the number of absolutely independent triples for non-binary trees is more complex, depending on the degree and level of resolution of the tree." Considering even a simple example of a non-binary tree, one sees that one rooted

triple for each internal edge is not enough to identify a tree. The triples 12|4, 13|5, 56|3, distinguish the three internal branches of the tree in figure 6.1 (adapted from [Grünewald et al., 2007, fig. 1]), but the tree with non trivial clusters $\{1, 2\}$, $\{1, 2, 3\}$, and $\{4, 5, 6\}$ also displays these triples and is not a resolution of the tree in figure 6.1. How many more rooted triples are required for a non-binary tree, however, is not immediately evident. We will establish a lower bound on the number of rooted triples needed to identify a tree, and show that this lower bound can be obtained for any tree. First we define a few graphs that are variants of the graph used in the Aho algorithm; these graphs simplify the proof of our main result.

Let R be a set of rooted triples, and let V and U be sets of subsets of $L(R)$. We define an edge-labeled graph $G(R, V, U)$ as follows. Take the vertices of the graph to be the elements of V. Add an edge between two vertices v and v' if $\exists\ ab|c \in R$ such that $a \in v$, $b \in v'$, and $c \in u$ for some $u \in U$. Label each edge $\{v, v'\}$ with the set $\{u \in U : \exists\ ab|c \in R$ such that $a \in v$, $b \in v'$, and $c \in u\}$. If $V = U = \{\{x\} : x \in L(R)\}$, then $G(R, V, U)$ is simply $G(R)$ with edge labels as defined in Bryant and Steel [1995].

Let T be a tree with $L(T) \subseteq L(R)$, and let $(u, v) \in E^\circ(T)$, where $E^\circ(T)$ is the set of interior edges of T. Then $G(R, T, (u, v)) := G(R, V, U)$, where $V = \{\mathrm{des}(x) : (v, x) \in E(T)\}$ and $U = \{\mathrm{des}(w) : (u, w) \in E(T),\ w \neq v\}$. Furthermore, for a vertex w of T such that $(u, w) \in E^\circ(T)$, and $w \neq v$, we use $G_w(R, T, (u, v))$ to denote the subgraph of $G(R, T, (u, v))$ with the same vertex set and only those edges which have $\mathrm{des}(w)$ in their label set.

Lemma 6.3.2. *If R is a set of rooted triples and $(u, v) \in E^\circ(\mathcal{A}_R)$, then $G(R, \mathcal{A}_R, (u, v))$ is connected.*

Proof. By the construction of $\mathcal{A}_R$, $\mathrm{des}(v)$ must be the vertex set of a connected component of $G(R|_{\mathrm{des}(u)})$. We will show that, ignoring edge labels, $G(R, \mathcal{A}_R, (u, v))$ can be obtained from $G(R|_{\mathrm{des}(u)})|_{\mathrm{des}(v)}$ by simply identifying vertices. Let G^* be

the graph obtained from $G(R|_{\text{des}(u)})|_{\text{des}(v)}$ by identifying all vertices that are in the same connected component of $G(R|_{\text{des}(v)})$. By the construction of $\mathcal{A}_R$, G^* and $G(R, \mathcal{A}_R, (u, v))$ have the same vertex set, say $B_1, B_2, ...B_{d_{out}(v)}$, where $d_{out}(v)$ denotes the out degree of v in $\mathcal{A}_R$. If $\{B_i, B_j\}$ is an edge of $G(R, \mathcal{A}_R, (u, v))$, then $\exists\ a \in B_i$, $b \in B_j$, and $c \in \text{des}(u)$ such that $ab|c \in R$. Therefore, $\{B_i, B_j\}$ is also an edge of G^*. Suppose that $\{B_i, B_j\}$ is an edge of G^* but is not in the edge set of $G(R, \mathcal{A}_R, (u, v))$. Then there must be some $a \in B_i$, $b \in B_j$, and $c \in \text{des}(u)$ such that $ab|c \in R$ and $c \notin \text{des}(u) - \text{des}(v)$. Thus $c \in \text{des}(v)$. This contradicts the fact that a and b are in distinct connected components of $G(R|_{\text{des}(v)})$. Therefore, G^* and $G(R, \mathcal{A}_R, (u, v))$ have the same edge set and we can conclude that $G(R, \mathcal{A}_R, (u, v))$ must be connected. □

Lemma 6.3.3. *If R is a set of rooted triples that identifies $\mathcal{A}_R$, then $\forall (u, v) \in E^\circ(\mathcal{A}_R)$ and $\forall (u, w) \in E(\mathcal{A}_R)$ such that $w \neq v$, $G_w(R, \mathcal{A}_R, (u, v))$ is connected.*

Proof. If $d_{out}(u) = 2$, then $G_w(R, \mathcal{A}_R, (u, v)) = G(R, \mathcal{A}_R, (u, v))$. Thus, by Lemma 6.3.2, $G_w(R, \mathcal{A}_R, (u, v))$ is connected.

Now consider the case where $d_{out}(u) > 2$. Suppose $G_w(R, \mathcal{A}_R, (u, v))$ has more than one connected component, and let $C_1, ..., C_k$ be its connected components with more than one vertex. If all vertices of $G_w(R, \mathcal{A}_R, (u, v))$ are isolated, i.e. if $k = 0$, let T be the tree obtained from $\mathcal{A}_R$ by simply replacing (u, w) with (v, w). Otherwise, let T be the tree obtained from $\mathcal{A}_R$ by replacing (u, w) with (v, w) and for each $i \in \{1, ..., k\}$, adding a vertex x_i, adding an edge (v, x_i), and adding an edge (x_i, y) for every y such that $des(y) \in C_i$. Since T is not a resolution of $\mathcal{A}_R$, and we assumed that R identifies $\mathcal{A}_R$, T must not display R. Then, there is a rooted triple $ab|c \in R$ which is displayed by A_R that is not displayed by T. This implies that $c \in des(w)$, and a and b are in distinct connected components of $G_w(R, \mathcal{A}_R, (u, v))$, which contradicts the definition of that graph. Therefore, $G_w(R, \mathcal{A}_R, (u, v))$ is connected. □

Theorem 6.3.4. *Given a tree T, and a set of rooted triples R, $\mathcal{A}_R = T$ if and only if the following two conditions hold:*

(i) $R \subseteq r(T)$,*and*

(ii) $\forall (u,v) \in E^\circ(T)$, $G(R,T,(u,v))$ *is connected.*

Furthermore, R identifies T if and only if, in addition to (i) and (ii), the following condition holds.

(iii) $\forall (u,v) \in E^\circ(T)$ *and for each* $(u,w) \in E(T)$ *with* $w \neq v$, $G_w(R,T,(u,v))$ *is connected.*

Proof. Assume that $\mathcal{A}_R = T$. Then $R \subseteq r(\mathcal{A}_R) = r(T)$, satisfying condition (i). By Lemma 6.3.2, condition (ii) must also be satisfied.

To prove the converse (i.e. conditions (i) and (ii) imply $\mathcal{A}_R = T$), we will induct on the number of internal edges of T. The result clearly holds for trees with only one internal edge. Let T be a tree with $|E^\circ(T)| \geq 1$. Assume the result holds for any tree with less than $|E^\circ(T)|$ leaves.

We assume that $\mathcal{A}_R \neq T$. Hence, there are edges (u,v) of T and (u',v') of $\mathcal{A}_R$ such that $des_T(u) = des_{\mathcal{A}_R}(u')$ and $des_T(v) \neq des_{\mathcal{A}_R}(v')$ and $des_T(v) \cap des_{\mathcal{A}_R}(v') \neq \emptyset$. Let T_v be the subtree of T with root v and leafset $des(v)$. Clearly, we have $R|_{des(v)} \subseteq r(T_v)$ and that $G(R_{des(v)}, T_v, (w_1,w_2))$ is connected for every edge $(w_1,w_2) \in E^\circ(T_v)$. By the induction hypothesis, we have $T_v = \mathcal{A}_{R|_{des(v)}}$. Hence, for every edge (v,w) of T, $des(w)$ is contained in one connected component of $G(R|_{des(v)})$ and therefore in one connected component of $G(R|_{des(u)})$. Further, since $G(R,T,(u,v))$ is connected, even $des(v)$ is contained in one connected component of $G(R|_{des(u)}) = G(R|_{des(u')})$; thus we have $des(v) \subseteq des(v')$. Since $des(v) \neq des(v')$, there is an edge $\{x_1,x_2\}$ in the connected graph $(R|_{des(u)})|_{des(v')}$ with $x_1 \in des(v)$ and $x_1 \notin des(v)$. Hence there is $y \in des(u)$ with $x_1x_2|y \in R$, but this rooted triple

is not displayed by T, a contradiction. This proves that $T = \mathcal{A}_R$, hence the first result of this theorem.

Now we will prove the second result, that conditions (i)-(iii) are necessary and sufficient for R to identify T. If R identifies T, then $T = \mathcal{A}_R$ [Semple, 2003]. Thus, conditions (i) and (ii) follow from the first part of this theorem and condition (iii) follows from Lemma 6.3.3.

Assume that conditions (i)-(iii) hold. By the first result in this theorem, $T = \mathcal{A}_R$. Suppose that R does not identify T. Then by a result in Daniel [2003, p. 45], $\exists\ (u,v),(u,w) \in E(T)$ such that $v \neq w$ and $G(R|_{\mathrm{des}(v)\cup\mathrm{des}(w)})$ has more than two connected components. We know that for each connected component C of $G(R|_{\mathrm{des}(v)\cup\mathrm{des}(w)})$, either $V(C) \subseteq \mathrm{des}(v)$ or $V(C) \subseteq \mathrm{des}(w)$. Assume without loss of generality that the vertices of at least two of the connected components of $G(R|_{\mathrm{des}(v)\cup\mathrm{des}(w)})$ have vertex sets which are subsets of $\mathrm{des}(v)$. Then the graph $G(R|_{\mathrm{des}(v)\cup\mathrm{des}(w)})$ cannot be connected, which contradicts the assumption that condition (iii) holds. Therefore, R must identify T.

□

Notice that the graphs in conditions (ii) and (iii) in Theorem 6.3.4 depend only on those rooted triples which distinguish an edge. Therefore, the following corollary is immediate.

Corollary 6.3.5. *If R is a minimal set of rooted triples identifying T, then each element of R distinguishes exactly one internal edge of T.*

We are now ready to establish a lower bound on the number of rooted triples needed to identify a tree. For a rooted phylogenetic tree T, we define

$$lb(T) = \sum_{(u,v)\in E^\circ(T)} (d_{out}(u) - 1) \cdot (d_{out}(v) - 1).$$

Theorem 6.3.6. *If R is a set of rooted triples that identifies T, then $|R| \geq lb(T)$.*

Proof. Let R be a minimal set of rooted triples that identifies T. By Corollary 6.3.5, each element of R distinguishes exactly one internal edge of T. For each internal edge (u, v) of T, let $\pi_{(u,v)}$ be the set of elements of R that distinguish (u, v). Then $\{\pi_{(u,v)} : (u, v) \in E^\circ(T)\}$ is a partition of R and

$$|R| = \sum_{(u,v)\in E^\circ(T)} |\pi_{(u,v)}|.$$

We will show that for every internal edge $(u, v) \in T$,

$$|\pi_{(u,v)}| \geq (d_{out}(u) - 1) \cdot (d_{out}(v) - 1).$$

Let (u, v) be an internal edge of T and let $W = \{w : (u, w) \in E(T), w \neq v\}$. By Lemma 6.3.3, for each $w \in W$, $G_w(R, T, (u, v))$ is connected. For each $w \in W$, let $G_w = G_w(R, T, (u, v))$; then,

$$|\pi_{(u,v)}| \geq \sum_{w\in W} |E(G_w)| \geq \sum_{w\in W} |V(G_w)|-1 = \sum_{w\in W} d_{out}(v)-1 = (d_{out}(u)-1)\cdot(d_{out}(v)-1).$$

This establishes the lower bound on $|R|$. □

Now we will show that for any tree T, we can choose a set R of rooted triples such that R identifies T and $|R| = lb(T)$.

Theorem 6.3.7. *For every phylogenetic tree T, there is a set R of rooted triples such that R identifies T and $|R| = lb(T)$.*

Proof. We prove this theorem by constructing such a set R. For each $(u, v) \in E^\circ(T)$, we choose a set of rooted triples $\pi_{(u,v)}$ in the following manner. Let $w_1, w_2, ..., w_{d_{out}(v)}$ be the children of v and, for $i \in \{1, 2, ..., d_{out}(v)\}$, let $x_i \in \text{des}(w_i)$ In addition, let $y_1, y_2, ..., y_{d_{out}(u)}$ be the children of u with $y_{d_{out}(u)} = v$ and, for $i \in \{1, 2, ..., d_out(u)-$

1}, let $z_i \in \text{des}(y_i)$. Then let

$$\pi_{(u,v)} = \{x_i x_{i+1} | z_j : 1 \leq i \leq d_{out}(v) - 1 \text{ and } 1 \leq j \leq d_{out}(u) - 1\}.$$

Finally, let

$$R = \bigcup_{(u,v) \in e^\circ(T)} \pi(u, v).$$

Then by its construction, R satisfies conditions (i)–(iii) of Theorem 6.3.4, thus R identifies T and $|R| = lb(T)$. □

6.4 Application to Existing Supertree Methods

Wilkinson et al. [2004] proposed a variant of MRP that, instead of each bipartition, each rooted triple in each source tree contributes a partial binary character to the matrix representation. As with MRP, the matrix representation is then analyzed under maximum parsimony to obtain a supertree. As Wilkinson et al. [2004] point out, this biases the outcome towards the relationships contained in larger source trees over those in smaller source trees. Wilkinson et al. suggest weighting the characters in the matrix representation to correct this bias. Specifically, each character obtained from a rooted triple contained in a tree T would be given a weight equal to "the ratio of the minimal number of [rooted triples] needed to specify the tree, divided by the number of [rooted triples] in the tree"[Wilkinson et al., 2004, p. 997]. They refer to such a weighting scheme as minimum fractional weighting (MFW), and give the weight $\frac{6}{n(n-1)}$ that would be assigned to a rooted triple from a binary tree with n leaves.

They go on to say that calculating the number of rooted triples needed to specify a non-binary tree "is more complex, depending on the degree and level of resolution in the tree. Based on Theorem 6.3.6, we can now know that the MFW

for a binary or non-binary rooted phylogenetic tree T is given by $\frac{lb(T)}{r(T)}$.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Many biological questions can only be answered in the light of evolution, and the tool most often used to provide that evolutionary context, for macro-evolutinary questions, is the phylogenetic tree. We have seen that the current state of both tree inference methods and availability of molecular sequence data means that, in order to construct some phylogenies, we need supertree methods. This dissertation has taken us through some issues with current supertree methods and presented new and better options for constructing them.

At the end of Chapter 2, I discussed the current state of supertree simulation studies and why more realistic simulations are needed. In Chapter 3, I presented a software package that incorporates biological and systematic factors into the production of simulated datasets for use in testing the performance of supertree methods. The two most important gains of this supertree simulation platform over those of previous simulation studies are 1) more complex models of gene evolution that include gene gain and loss over the model tree and 2) a selection process for source tree datasets that reflects the practices of systematists.

In Chapter 4, I used this simulation platform to test MRP, the most broadly used supertree method, and its variant, weighted MRP, against each other and combined analysis. Our findings were different than those of Bininda-Emonds and Sanderson [2001], and are the first that show that combined analysis consistently, and often significantly, outperforms MRP and wMRP. It is possible that these differences were due to our use of more realistic datasets. In our simulation studies, MRP approached the topological accuracy of combined analysis only when the source trees included a full tree, a condition that is less likely to encourage a researcher to turn to supertree methods. The results in this chapter clearly demonstrate the need for better supertree methods.

In Chapter 5, I presented such a method. I described SuperFine and showed that its performance was superior to that of MRP and that it approached the accuracy of a combined analysis. Furthermore, SuperFine runs in significantly less time than MRP, an advantage that increases as the number of taxa increases. These properties make SuperFine a very attractive supertree method, particularly when no complete source tree is available. I am excited about the possible uses for SuperFine, both as a supertree method and as a consensus method that can be used to perform ML analyses on a typical desktop computer that would otherwise require a super-computer.

In Chapter 6, I turned to more theoretical pursuits. I discussed sets of rooted triples and their closures, and determined the minimum number of rooted triples needed to uniquely define a given rooted phylogenetic tree. I also showed how this result could be used in a proposed variant of the MRP supertree method, which until now could only be used on binary source trees.

7.2 Future Work

I have already shown in this dissertation that SuperFine is almost as accurate as combined analysis, but could it be better? There are several aspects of the algorithm design space that could be explored as possible ways to improve the accuracy and running time of SuperFine:

- The supertree method used in the first phase of the algorithm, obtaining an unresolved supertree whose polytomies will subsequently be resolved in the second phase of SuperFine, deserves exploration. One could use any supertree method for this phase, but ideally this method should return a supertree with a low false positive rate. Gordon's strict consensus merger, which returns a supertree that is a restriction of the SCM supertree, is a natural candidate. One could also try using the strict consensus of the most parsimonious, or even the near-most parsimonious, trees found in an MRP analysis.

- The technique used to obtain quartet encodings of the source trees could potentially be refined. For a given polytomy v of degree d, the current implementation of SuperFine produces a quartet encoding of the source trees by first relabeling the leaves by elements in $\{1, ..., d\}$, then collapsing the subtree with leaves labeled i to a single leaf labeled i, and finally taking the full set of induced quartet trees for every relabeled and collapsed source tree. One could obtain a different quartet encoding by varying the second and third of these three steps. Eliminating the collapsing step and/or changing the method used to extract quartet trees from the relabeled (and possibly collapsed) source trees, would alter the set of quartet trees handed to QMC, and hence the tree used to resolve v.

  Another adjustment to this second step that might improve the performance, in terms of accuracy, memory usage, and running time, of SuperFine is to

create a version of QMC that takes weighted quartet trees. QMC currently takes a multi-set of quartet trees (i.e., it allows for multiple copies of the same quartet tree), and thus essentially allows for integer weights on the quartet trees. Modifying QMC to take sets of weighted quartet trees would allow for a wider range of weights as well as reduce memory usage and possibly running time for that method.

- Many of the processes performed in the SuperFine algorithm could be run in parallel. The most obvious candidate is the second phase, the resolution of polytomies. Since resolving one polytomy is not affected by the resolution of any other polytomies, these resolutions could be performed in parallel. Potential parallelization could also take place in the computation of the SCM supertree.

In addition to exploring ways to improve the performance of SuperFine as a supertree method, one could explore ways to use SuperFine to enable other types of phylogeny estimation:

- SuperFine could be used to construct a phylogeny without first constructing an alignment. In Section 2.3, I described the four phases of a typical phylogenetic analysis: data collection, sequence alignment, tree inference, and post-processing. For very large datasets, upwards of 50,000 taxa, the second phase, computing an alignment, could be impractical due to memory and running time limitations. SuperFine could be used in a divide-and-conquer approach, computing alignments only on relatively small subproblems, estimating trees on those sub-alignments, and using SuperFine to combine the subtrees into a single tree on the original, full set of taxa.

- SuperFine could also be used essentially as a consensus method, to produce a tree on a many-marker combined dataset where the total sequence length

makes a combined analysis infeasible or impossible. In this case, one could compute a possibly full tree on each marker, and use SuperFine to combine these estimates into a single tree. Any consensus method, e.g., strict or majority consensus, could of course be used in this combining step; however SuperFine would likely return a fully, or near fully, resolved consensus tree.

SMIDGen also has potential uses in future work. My use of SMIDGen to enable the simulation study presented in Chapter 4 was a big step towards evaluating the performance of supertree methods under realistic conditions, however there are several characteristics of empirical supertree studies that were not incorporated in this study:

- In Chapter 4, I showed that the density of the scaffold tree significantly impacted the accuracy of supertree methods. There are other characteristics of the scaffold whose impact remains untested, including, most notably, the resolution of the scaffold tree. Many empirical supertree studies include a 100% scaffold in the form of a taxonomy and thus, in these cases, the scaffold is highly unresolved.

- In the simulation studies presented in Chapters 4 and 5 of this dissertation, each clade-based study was based on a distinct clade. However, in empirical supertree studies, there can be multiple source trees focused on a single group. As a result, many empirical supertree studies have many more source trees than were studied here. Having more source trees could improve the performance of SuperFine in particular, because more source trees would inevitably lead to a lower false positive rate in the SCM tree obtained in the first phase of SuperFine.

- Finally, some recent empirical supertrees have been computed by taking supertree of supertrees.[e.g., Bininda-Emonds et al., 2007] The accuracy of the

trees produced via such procedures is completely unexplored and SMIDGen could easily be used to test such procedures.

Appendix

Tree Reconstruction Commands

Parsimony ratchet comands.– The PAUP* block used to perform the parsimony ratchet search is as follows:

```
begin paup;
set autoclose=yes warntree=no warnreset=no notifybeep=no
monitor=yes taxlabels=full;
log file=logfile replace;
set criterion=parsimony; pset collapse=no;
              [!][!*** Replicate 0 (initial tree) ***]
      hsearch addseq=random nreps=1 rseed=<random_seed> swap=TBR
multrees=no dstatus=60;
savetrees file=<tree_file> format=altnex replace;
       Then, for each integer i from 1 to n, [!][!*** Replicate i ***]
       weights list of character weights (chosen as described in Methods section);
hsearch start=current swap=TBR multrees=no dstatus=60;
weights 1:all;
hsearch start=current swap=TBR
multrees=no dstatus=60;
savetrees file=<tree_file> format=nexus append;
           [!][!*** Determining strict consensus tree ***]
```

```
set MaxTrees=<number_of_replicates>gettrees file=<tree_file>
allblocks=yes warntree=no;
contree all / strict=yes majrule=yes treefile=contreefile replace;
tcondense collapse=no deldupes=yes;
savetrees file=treefile replace=yes format=altnex;
log stop; end; quit warntsave=no;
```

Maximum parsimony bootstrap commands.– The PAUP* commands used in MP bootstrap analysis are as follows:

```
begin paup;
set criterion=parsimony maxtrees=1000 increase=no storetreewts=yes;
bootstrap bseed=<random_seed> nreps=1000 search=faststep
treefile=<tree_output_file> replace=yes;
savetrees file=<consensus_output_file> replace=yes
savebootp=brlens from=1 to=1 format=altnex;
gettrees file=<tree_output_file> storetreewts=yes mode=3;
savetrees file=<output_file> replace=yes format=phylip;
contree /majrule=yes strict=no usetreewts=yes
treefile=<consensus_output_file> replace;
quit; end;
```

Weighted maximum parsimony search commands.– The PAUP* commands used in weighted MP analysis are as follows:

```
begin paup;
set criterion=parsimony maxtrees=1000 increase=no;
hsearch start=stepwise addseq=random nreps=100 swap=tbr;
lter best=yes;
savetrees file=<output_file> replace=yes format=altnex;
contree all/ strict=yes majrule=yes
```

```
treefile=<consensus_output_file>replace=yes;
quit; end;
```

Maximum likelihood commands.– RAxML commands used in ML analyses are as follows:

For ML source trees and 100- and 500-taxon CA-ML analysis: `raxmlHPC -s <phylip_alignment_file> -n <output_suffix> -m GTRMIX -w <working_directory>`

For 1000-taxon CA-ML analysis: `raxmlHPC -s <phylip_alignment_file> -n <output_suffix> -m GTRCAT -w <working_directory>`

Maximum likelihood boostrap commands.– RAxML commands used in ML bootstrap analysis are as follows:

`raxmlHPC -s <phylip_alignment_file> -n <output_suffix> -m GTRMIX -w <working_directory>`
(computes a ML tree which we will annotate with bootstrap values)

`raxmlHPC -s <phylip_alignment_file> -n <bootstrap_analysis_output_suffix> -m GTRMIX -# 100 -b <random_seed> -w <working_direcory>`
(computes ML trees from 100 bootstrap replicates)

`raxmlHPC -f b -s <phylip_alignment_file> -n <bootstrap_tree_output_suffix> -m GTRMIX -z <boostrap_output_file> -t <ml_tree_file> -w <working_directory>`
(annotates the tree computed in the first step with bootstrap values)

Bibliography

J. Baldwin, D. Fitch, P. De Ley, and S. Nadler. The Nematode Branch of the Assembling the Tree of Life Project: NemATOL, 2008. http://nematol.unh.edu.

B. R. Baum. Combining trees as a way of combining data sets for phylogenetic inference, and the desirability of combining gene trees. *Taxon*, 41:3–10, 1992.

B. R. Baum and M. A. Ragan. The MRP method. In O. R. P. Bininda-Emonds, editor, *Phylogenetic Supertrees: Combining Information To Reveal The Tree Of Life*, pages 17–34. Kluwer Academic, Dordrecht, the Netherlands, 2004.

R. M. D. Beck, O. R. P. Bininda-Emonds, M. Cardillo, F. G. R. Liu, and A. Purvis. A higher-level mrp supertree of placental mammals. *BMC Evol. Biol.*, 6:93, 2006.

O. R. P. Bininda-Emonds. MRP supertree construction in the consensus setting. In *Bioconsensus*, volume 61 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 231–242. American Mathematical Society-DIMACS, Providence, Rhode Island, 2003.

O. R. P. Bininda-Emonds and M. J. Sanderson. Assessment of the accuracy of matrix representation with parsimony analysis supertree construction. *Syst. Biol.*, 50:565–579, 2001.

O. R. P. Bininda-Emonds, J. L. Gittleman, and A. Purvis. Building large trees

by combining phylogenetic information: a complete phylogeny of the extant Carnivora (Mammalia). *Biol. Rev. Camb. Philos. Soc.*, 74:143–175, 1999.

O. R. P. Bininda-Emonds, K. E. Jones, S. A. Price, R. Grenyer, M. Cardillo, M. Habib, A. Purvis, and J. L. Gittleman. Supertrees are a necessary Not-so-Evil: a comment on Gatesy et al. *Syst. Biol.*, 52(5):724–729, 2003.

O. R. P. Bininda-Emonds, M. Cardillo, K. E. Jones, R. D. E. MacPhee, R. M. D. Beck, R. Grenyer, S. A. Price, R. A. Vos, J. L. Gittleman, and A. Purvis. The delayed rise of present-day mammals. *Nature*, 446:507–512, 2007.

R. S. Boyer, W. A. Hunt, and S. M. Nelesen. A compressed format for collections of phylogenetic trees and improved consensus performance. In *In Algorithms in Bioinformatics: 5th International Workshop, WABI 2005, number 3692 in Lecture Notes in Computer Science*, volume 3692, pages 353–364, 2005.

M. J. Brauer, M. T. Holder, L. A. Dries, D. J. Zwickl, P. O. Lewis, and D. M. Hillis. Genetic algorithms and parallel processing in maximum-likelihood phylogeny inference. *Mol. Biol. Evol.*, 19:1717–1726, 2002.

D. Bryant and M Steel. Extension operations on sets of leaf-labelled trees. *Adv. in Appl. Math.*, 16:425–453, 1995.

J. G. Burleigh, O. Eulenstein, D. Fernández-Baca, and M. J. Sanderson. MRF supertrees. In O. R. P. Bininda-Emonds, editor, *Phylogenetic Supertrees: Combining Information To Reveal The Tree Of Life*, pages 65–86. Kluwer Academic, Dordrecht, the Netherlands, 2004.

R. M. Bush, C. A. Bender, K. Subbarao, N. J. Cox, and W. M. Fitch. Predicting the evolution of human influenza a. *Science*, 286(5446):1921–1925, 1999.

M. Cardillo, O. R. P. Bininda-Emonds, E. Boakes, and A. Purvis. A species-level phylogenetic supertree of marsupials. *J. Zool.*, 264:11–31, 2004.

M. J. Cavalcanti. A phylogenetic supertree of the hammerhead sharks (Carcharhiniformes: Sphyrnidae). *Zool. Stud.*, 46:6–11, 2007.

M. W. Chase, D. E. Soltis, R. G. Olmstead, D. Morgan, D. H. Les, B. D. Mishler, M. R. Duvall, R. A. Price, H. G. Hills, and Y. L. Qiu. Phylogenetics of seed plants: An analysis of nucleotide sequences from the plastid gene rbcL. *Ann. Mo. Bot. Garden*, 80:528–580, 1993.

D. Chen, L. Diao, O. Eulenstein, D. Fernández-Baca, and M. J. Sanderson. Flipping: A supertree construction method. In *Bioconsensus*, volume 61 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, pages 135–160. American Mathematical Society-DIMACS, Providence, Rhode Island, 2003.

D. Chen, O. Eulenstein, D. Fernández-Baca, and J. G. Burleigh. Improved heuristics for minimum-flip supertree construction. *Evol. Bioinform.*, 2:401–410, 2006.

B. Chor and T. Tuller. Maximum likelihood of evolutionary trees: hardness and approximation. *Bioinformatics*, 21(1):97–106, 2005.

A. Criscuolo, V. Berry, E. Douzery, and O. Gascuel. SDM: A fast distance-based approach for (super) tree building in phylogenomics. *Syst. Biol.*, 55:740–755, 2006.

P. Daniel. Supertree methods: some new approaches. Master's thesis, Universith of Canterbury, New Zealand, 2003.

V. Daubin, M. Gouy, and G. Perriere. Bacterial molecular phylogeny using supertree approach. *Genome Inform.*, 12:155–164, 2001.

T. J. Davies, T. G. Barraclough, M. W. Chase, P. S. Soltis, D. E. Soltis, and V. Savolainen. Darwin's abominable mystery: Insights from a supertree of the angiosperms. *Proc. Natl. Acad. Sci.*, 101:1904–1909, 2004.

W. H. E. Day. Optimal algorithms for comparing trees with labeled leaves. *J. Classif.*, 2(1):7–28, 1985.

O. Eulenstein, D. Chen, J. G. Burleigh, D. Fernández-Baca, and M. J. Sanderson. Performance of flip supertree construction with a heuristic algorithm. *Syst. Biol.*, 53:299–308, April 2004.

J. Felsenstein. *Inferring Phylogenies.* Sinauer Associates, Sunderland, Massachusetts, 2003.

W. M. Fitch. Toward defining the course of evolution: minimal change for a specific tree topology. *Syst. Zool.*, 20:406–416, 1971.

L. R. Foulds and R. L. Graham. The steiner problem in phylogeny is NP-complete. *Adv. in Appl. Math.*, 3(43-49):299, 1982.

G. Ganapathy. *Algorithms and Heuristics for Combinatorial Optimization in Phylogeny.* PhD thesis, University of Texas at Austin, 2006.

J. Gatesy and M. S. Springer. A critique of matrix representation with parsimony supertrees. In O. R. P. Bininda-Emonds, editor, *Phylogenetic Supertrees: Combining Information To Reveal The Tree Of Life*, pages 369–388. Kluwer Academic, Dordrecht, the Netherlands, 2004.

J. Gatesy, C. Matthee, R. DeSalle, and C. Hayashi. Resolution of a supertree/supermatrix paradox. *Syst. Biol.*, 51:652–664, 2002.

H. Glenner, A. J. Hansen, M. V. Sørensen, F. Ronquist, J. P. Huelsenbeck, and E. Willerslev. Bayesian inference of the metazoan phylogeny: A combined molecular and morphological approach. *Current Biol.*, 14:1644–1649, 2004.

P. A. Golobof and D. Pol. Semi-strict supertrees. *Cladistics*, 18:514–525, 2002.

R. Grenyer and A. Purvis. A composite species-level phylogeny of the 'Insectivora(Mammalia: Order Lipotyphla Haeckel, 1866). *J. Zool.*, 260:245–257, 2003.

S. Grünewald, M. Steel, and M. S. Swenson. Closure operations in phylogenetics. *Math. Biosci.*, 208(2):521–537, 2007.

J. Hartigan. Minimum mutation fits to a given tree. *Biometrics*, 29:53–65, 1972.

T. A. Heath, S. M. Hedtke, and D. M. Hillis. Taxon sampling and the accuracy of phylogenetic analyses. *J. Syst. and Evol.*, 46(3):239–257, 2008.

B. Holland, G. Conner, K. Huber, and V. Moulton. Imputing supertrees and supernetworks from quartets. *Syst. Biol.*, 57(1):299–308, 2007.

D. H. Huson, S. M. Nettles, and T. J. Warnow. Disk-Covering, a fast-converging method for phylogenetic tree reconstruction. *J. Comput. Biol.*, 6(3-4):369–386, 1999a.

D. H. Huson, L. Vawter, and T. Warnow. Solving large scale phylogenetic problems using DCM2. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology table of contents*, pages 118–129. AAAI Press, 1999b.

K. E. Jones, A. Purvis, A. MacLarnon, O. R. P. Bininda-Emonds, and N. B. Simmons. A phylogenetic supertree of the bats (Mammalia: Chiroptera). *Biol. Rev. Camb. Philos. Soc.*, 77:223–259, 2002.

T. H. Jukes and C. Cantor. *Mammalian Protein Metabolism.* Academic Press, New York, 1969.

M. Kennedy and R. D. M. Page. Seabird supertrees: combining partial estimates of procellariiform phylogeny. *The Auk*, 119:88–108, 2002.

J. Kim and T. Warnow. Tutorial on phylogenetic tree estimation. 1999. In *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology (ISMB99)*, 1999.

M Kimura. A simple method for estimating evolutionary rate of base substitutions through comparative studies of nucleotide sequences. *J. Mol. Evol.*, 16:111–120, 1980.

F. J. Lapointe and C. Levasseur. Everything you always wanted to know about average consensus and more. In O. R. P. Bininda-Emonds, editor, *Phylogenetic Supertrees: Combining Information To Reveal The Tree Of Life*, pages 87–106. Kluwer Academic, Dordrecht, the Netherlands, 2004.

C. R. Linder and T. Warnow. An overview of phylogeny reconstruction. In Srinivas Aluru, editor, *Handbook of Computational Molecular Biology*, pages 19–1–19–40. Chapman & Hall/CRC, Taylor & Francis Group, 2006.

F. G. R. Liu, M. M. Miyamoto, N. P. Freire, P. Q. Ong, M. R. Tennant, T. S. Young, and K. F. Gugel. Molecular and morphological supertrees for eutherian (placental) mammals. *Science*, 291:1786–1789, 2001.

M. L. Metzker, D. P. Mindell, X. M. Liu, R. G. Ptak, R. A. Gibbs, and D. M. Hillis. Molecular evidence of HIV-1 transmission in a criminal case. *Proceedings of the National Academy of Sciences*, 99(22):14292–14297, 2002.

L. Nakhleh, U. Roshan, K. St. James, J. Sun, and T. Warnow. Designing fast converging phylogenetic methods. *Bioinformatics*, 17:190–198, 2001.

R. D. M. Page. Taxonomy, supertrees, and the tree of life. In O. R. P. Bininda-Emonds, editor, *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*, pages 247–265. Kluwer Academic, Dordrecht, the Netherlands, 2004.

R. Piaggio-Talice, J. G. Burleigh, and O. Eulenstein. Quartet supertrees. In O. R. P. Bininda-Emonds, editor, *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*, pages 173–191. Kluwer Academic, Dordrecht, the Netherlands, 2004.

S. A. Price, O. R. P. Bininda-Emonds, and J. L. Gittleman. A complete phylogeny of the whales, dolphins and even-toed hoofed mammals (Cetartiodactyla). *Biol. Rev. Camb. Philos. Soc.*, 80:445–473, 2005.

A. Purvis. A composite estimate of primate phylogeny. *Philos. Trans. R. Soc. London [Biol.]*, 348:405–421, 1995a.

A. Purvis. A modification to Baum and Ragan's method for combining phylogenetic trees. *Syst. Biol.*, 44:251–255, 1995b.

M. A. Ragan. Phylogenetic inference based on matrix representation of trees. *Mol. Phylo. Evol.*, 1:53–58, 1992.

A. G. Rodrigo. A comment on Baum's method for combining phylogenetic trees. *Taxon*, 42:631–636, 1993.

F. Ronquist. Matrix representation of trees, redundancy, and weighting. *Syst. Biol.*, 45:247–253, June 1996.

U. Roshan, B. M. E. Moret, T. L. Williams, and T. Warnow. Performance of supertree methods on various dataset decompositions. In O. R. P. Bininda-Emonds, editor, *Phylogenetic Supertrees: Combining Information To Reveal The Tree Of Life*, pages 301–328. Kluwer Academic, Dordrecht, the Netherlands, 2004a.

U. W. Roshan, T. Warnow, B. M. E. Moret, and T. L. Williams. Rec-I-DCM3: a fast algorithmic technique for reconstructing phylogenetic trees. In *2004 IEEE Computational Systems Bioinformatics Conference, 2004. CSB 2004. Proceedings*, pages 98–109, 2004b.

H. A. Ross and A. G. Rodrigo. An assessment of matrix representation with compatibility in supertree construction. In O. R. P. Bininda-Emonds, editor, *Phylogenetic Supertrees: Combining Information To Reveal The Tree Of Life*, pages 35–64. Kluwer Academic, Dordrecht, the Netherlands, 2004.

S. M. Ross, editor. *Stochastic Processes*. Wiley, New York, 2nd edition, 1996.

N. Salamin, T. R. Hodkinson, and V. Savolainen. Building supertrees: An empirical assessment using the grass family (Poaceae). *Syst. Biol.*, 51:136–150, 2002.

M. J. Sanderson. r8s: inferring absolute rates of molecular evolution and divergence times in the absence of a molecular clock. *Bioinformatics*, 19:301–302, 2003.

D. W. Schwilk and D. D. Ackerly. Flammability and serotiny as strategies: correlated evolution in pines. *Oikos*, 94:326, 2001.

C. Semple. Reconstructing minimal rooted trees. *Discr. Appl. Math.*, 127:489–503, 2003.

C. Semple and M. Steel. *Phylogenetics*. Oxford University Press, 2003.

J. B. Slowinski. How should species phylogenies be inferred from sequence data? *Syst. Biol.*, pages 814–825, 1999.

S. Snir and S. Rao. Quartets MaxCut: a divide and conquer quartets algorithm. In *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 2008.

D. E. Soltis, P. S. Soltis, D. L. Nickrent, L. A. Johnson, W. J. Hahn, S. B. Hoot, J. A. Sweere, R. K. Kuzoff, K. A. Kron, and M. W. Chase. Angiosperm phylogeny inferred from 18S Ribosomal DNA sequences. *Ann. Mo. Bot. Garden*, 84:1–49, 1997.

A. Stamatakis. RAxML-NI-HPC: Maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics*, 22:2688–2690, 2006.

M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *J. Classif.*, 9:91–116, 1992.

M. Steel and D. Penny. Parsimony, likelihood, and the role of models in molecular phylogenetics. *Mol. Biol. Evol.*, 17(6):839–850, 2000.

C. J. Stoner, O. R. P. Bininda-Emonds, and T. Caro. The adaptive significance of coloration in lagomorphs. *Biol. J. Linn. Soc.*, 79:309–328, 2003.

D. L. Swofford, G. J. Olson, P. J. Waddell, and D. M. Hillis. *Phylogenetic Inference*, pages 407–425. Sinauer Associates, Sunderland, Massachusetts, 2nd edition edition, 1996.

J. Tang and B. M. E. Moret. Scaling up accurate phylogenetic reconstruction from gene-order data. *Bioinformatics*, 19:305–312, 2003.

A. R. Templeton. Human origins and analysis of mitochondrial DNA sequences. *Science*, 255(5045):737–737, 1992.

D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: the condor experience. *Concurrency and Computation: Practice and Experience*, 17: 323–356, 2005.

J. D. Thompson, D. G. Higgins, and T. J. Gibson. ClustalW: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680, 1994.

T. Warnow, B. M. E. Moret, and K. S. John. Absolute convergence: true trees from short sequences. In *Proceedings of the twelfth annual ACM-SIAM symposium on discrete algorithms*, pages 186–195. Society for Industrial and Applied Mathematics Philadelphia, PA, USA, 2001.

M. Wilkinson, J. A. Cotton, and J. L. Thorley. The information content of trees and their matrix representations. *Syst. Biol.*, 53(6):989–1001, 2004.

M. Wilkinson, J. A. Cotton, C. Creevey, O. Eulenstein, S. R. Harris, F.-J. Lapointe, C. Levasseur, J. O. McInerney, D. Pisani, and J. L. Thorley. The shape of supertrees to come: Tree shape related properties of fourteen supertree methods. *Syst. Biol.*, 54(3):419–431, 2005.

M. F. Wojciechowski, M. J. Sanderson, K. P. Steele, and A. Liston. Molecular phylogeny of the "temperate herbaceous tribes" of papilionoid legumes: a supertree approach. *Adv. in Legume Syst.*, 9:277–298, 2000.

D. J. Zwickl. *Genetic algorithm approaches for the phylogenetic analysis of large biologiical sequence datasets under the maximum likelihood criterion*. PhD thesis, University of Texas at Austin, 2006.

Vita

Michelle Dawn Swenson was born to Pamela and William Swenson in Lincoln, Nebraska. Her passions include: first, dance, which she began studying at the age of three; second, math, a passion which she discovered only just before college; and third, teaching, initially teaching dance beginning at age 15 and subsequently teaching math, working as a TA during her undergraduate years at the University of Nebraska at Lincoln.

After earning her Bachelors of Science in Mathematics from the University of Nebraska at Lincoln, she entered the Ph.D. program in mathematics at the University of Texas at Austin. In the beginning of her graduate studies, she knew only that she wanted to apply math to biological questions. Her desire to apply discrete math to biological questions found an ideal outlet within the Integrative Graduate Education and Research Traineeship (IGERT) program, allowing her to work in collaboration with biologists and computer scientist within an environment of open exchange and mutual investigation. Within the IGERT program, computational phylogenetics provided a powerful subject for which her interest in applying graph theory to the study of biology made for an exciting and fruitful combination. Throughout her graduate work at the University of Texas at Austin, she continued to pursue teaching mathematics on the collegiate level, and also began leading undergraduate research projects. In parallel with her graduate career, she continued her study of dance, with a focus on ballet, performing first with Austin Dance

Ensemble and then, professionally, with Austin Classical Ballet.

Permanent Address: 3400 Speedway Apt. 304
Austin, TX 78705
USA

This dissertation was typeset with LaTeX 2_ε[1] by the author.

[1] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay, James A. Bednar, and Ayman El-Khashab.